Escuela
Politécnica
Superior

Sistema de inspección visual utilizando Integrated vision de ABB

Grado en Ingeniería Robótica

Trabajo Fin de Grado

Autor:
Eros Piera Moreno
Tutor:
Santiago Timoteo Puente Mendez

Julio 2021

Universitat d'Alacant
Universidad de Alicante

Sistema de inspección visual utilizando Integrated vision de ABB

Autor
Eros Piera Moreno

Tutor
Santiago Timoteo Puente Mendez
Departamento de Física de la Ingeniería de sistemas y Teoría de la señal

Grado en Ingeniería Robótica

Escuela Politécnica Superior

Universitat d'Alacant
Universidad de Alicante

ALICANTE, Julio 2021

Preámbulo

"La razón por la que se ha llevado a cabo este proyecto es el interés por la robótica industrial y su aplicación en la industria 4.0. El objetivo es poder realizar un proceso de detección y manipulación de piezas geométricas, mediante un sistema robótico compuesto por un brazo articulado y una cámara de visión integrada. "

Agradecimientos

Este trabajo no habría sido posible sin la ayuda de mi tutor Santiago Puente, el cual me propuso el tema del trabajo realizado y me ha apoyado durante las sesiones de laboratorio para poder desarrollar el proyecto.

También quería agradecer todo el apoyo que me ha ofrecido mi familia a lo largo de la realización del trabajo y de la carrera, me han llenado de energía día tras día hasta acabar esta etapa de mi vida.

No te ahogas al caer a un río,
sino al mantenerte sumergido en el

Paulo Coelho.

Índice general

Índice de figuras

Índice de Códigos

1. Introducción

En la actualidad se está comenzando a implantar la robótica industrial en las fábricas de producción, ya que ofrece un apoyo a los operarios automatizando gran parte de los procesos repetitivos y que pueden llegar a tener un coste a largo plazo en la salud de los trabajadores. En este proyecto se va a realizar una tarea de manipulación de piezas geométricas, lo cual podría ser perfectamente un trabajo automatizado en una planta de producción. La detección de las piezas se realizará con un sistema de visión llamado In-Sight 7402C que estará integrado en el robot IRB-120 de ABB, de esta manera es el robot el que recibirá los datos necesarios para poder obtener la posición de la figura en el espacio de trabajo y manejarla. La visión integrada tiene numerosas aplicaciones en la industria ya que puede mejorar la trazabilidad de los productos a lo largo de la cadena de suministro, detectar productos no conformes con las especificaciones requeridas y clasificar e identificar piezas.

Por lo que a lo largo de este proyecto, en la [Sección 2] se habla de la evolución de los robots manipuladores hasta la actualidad. En la [Sección 3], se comentan los diferentes objetivos propuestos durante la ejecución del proyecto junto con una serie de motivaciones personales. Seguidamente en la [Sección 4], se explicarán las herramientas utilizadas en el desarrollo del trabajo. A continuación, en la [Sección 5], se explicarán paso por paso los procedimientos que se han seguido para cumplir los objetivos propuestos, desde diseñar la herramienta de trabajo hasta la puesta en marcha del robot en el entorno real. En la [Sección 6], se comentarán los resultados obtenidos y se proporcionará un enlace para la visualización del trabajo de la estación en el entorno virtual y en el entorno real. Finalmente, en la [Sección 7] se expondrán los objetivos realizados y se comentará lo aprendido en el transcurso del trabajo. Para concluir, en la bibliografía, se adjuntan todos los enlaces utilizados con los que se ha obtenido la información necesaria para profundizar en cada sección. Además, en los anexos, se muestran los códigos desarrollados para realizar tanto la tarea de manipulación en el entorno de simulación [Anexo A.1] como la del entorno real [Anexo B.1].

2. Estado del arte

La robótica es una disciplina científica que combina la investigación y el desarrollo de unos sistemas mecánicos denominados robots, los cuales están diseñados para realizar tareas específicas o reducir la carga de trabajo en aplicaciones industriales, domésticas, científicas y comerciales. El término robot nace de la palabra checa *"robota"* que significa trabajo y apareció por primera vez en 1921, en la novela *Rossum's Universal Robots* escrita por el dramaturgo *Karel Capek* (1890-1938). Para más información (Cortés, 2011).

A mediados del siglo XX, la robótica empezó a desarrollarse debido a la necesidad de manipular materiales peligrosos, como sustancias radioactivas, venenos u objetos a altas temperaturas. En consecuencia, en 1948, apareció el primer sistema de teleoperación [Figura 2.1], creado por (*Raymond Goertz*, 1915-1970), para manipular elementos radioactivos. De esta manera, las aplicaciones de la robótica evolucionaron junto con la tecnología de los ordenadores, hasta que en 1954 (*George Devol*, 1912-2011) patentó el primer robot con memoria capaz de repetir movimientos. Más tarde, en 1961, *George Devol* fundó junto con (*Joseph Engelberger*, 1925-1915) lo que sería la primera compañía robótica de la historia *Unimation (Universal Automation).*

Figura 2.1: Raymond Goertz utilizando el sistema de teleoperación.

Entre los años 1960 y 1970, se establecieron las bases de la robótica, en concreto las bases de los diseños actuales de los robots industriales. Esto permitió a (*Victor Scheinman*, 1942-2016) inventar el Standford Arm [Figura 2.2a], el primer robot articulado de 6 ejes con accionamiento eléctrico controlado mediante un computador. Posteriormente, el *Dr. Victor Scheinman* vendió el diseño del Standford Arm a la compañía *Unimation*, a la que le sirvió como diseño base para desarrollar junto con la empresa *General Motors* el robot PUMA *(Programmable Universal Manipulation Arm)* [Figura 2.2b]. Éste fue el producto que lanzó el negocio de los robots manipuladores en Estados Unidos y Europa consiguiendo automatizar las líneas de producción.

(a) Standford Arm.

(b) Robot PUMA.

Figura 2.2: Robots Standford Arm y PUMA.

En la actualidad, existen compañías que tienen sus plantas de producción completamente automatizadas mediante robots manipuladores. Los avances tecnológicos en robótica y en dispositivos tecnológicos como sensores, se están aplicando a la industria, abriendo un escenario de enormes oportunidades que apuntan a la industria 4.0. Entre las tecnologías de la industria 4.0, se encuentra la robótica colaborativa. La cual incorpora en las líneas de producción un tipo de robot específico capaz de interactuar con operarios ofreciendo adaptabilidad, seguridad y flexibilidad en la producción, para más información (del Val Román, 2016).

3. Objetivos y motivación

La motivación para realizar este proyecto se basa en el desarrollo de una aplicación de detección y manipulación de piezas geométricas, utilizando un sistema robótico compuesto por un robot articular de 6 grados de libertad, un elemento terminal y un dispositivo de visión integrada. De esta manera, la motivación personal para realizar el trabajo es:

1. Mejorar el conocimiento general sobre la robótica industrial.
2. Profundizar en el uso de RobotStudio.
3. Conocer el proceso necesario para realizar la puesta en marcha de un sistema robótico.
4. Utilizar técnicas de visión por computador en el proyecto.
5. Mejorar en el diseño de piezas y escenarios 3D.

En consecuencia, los objetivos principales del proyecto son los siguientes:

1. Realizar una tarea de detección y manipulación de objetos.
2. Integrar en el sistema un dispositivo de visión.
3. Diseñar e integrar en el sistema un elemento terminal con accionamiento neumático.
4. Sincronizar todos los dispositivos que forman parte del sistema robótico.

4. Metodología

4.1. FreeCad

FreeCAD es una aplicación de modelado paramétrico 3D y la información utilizada para esta sección ha sido obtenida de (*FreeCAD Getting started*, 2007). La aplicación está diseñada principalmente para el modelado de piezas mecánicas o figuras geométricas como las que se van a diseñar posteriormente en este proyecto, pero también sirve para diseños de precisión y control del historial de modelado de las piezas. Es un software libre, por lo que existe una comunidad que está en continuo crecimiento, ayudando a mejorar el programa, corregir errores y ofreciendo muchos ejemplos de proyectos libres.

La interfaz del programa se puede visualizar en la siguiente [Figura 4.1]. El concepto principal detrás de la interfaz de FreeCAD, es que está separado en ambientes de trabajos. Un ambiente de trabajo es una colección de herramientas adecuadas para una tarea específica, como trabajar con mallas, dibujar objetos 2D o dibujar bocetos acotados. Además, se puede cambiar el ambiente de trabajo actual con el selector de banco de trabajo, permitiendo personalizar herramientas, agregarlas de otro ambiente de trabajo o incluso creándolas.

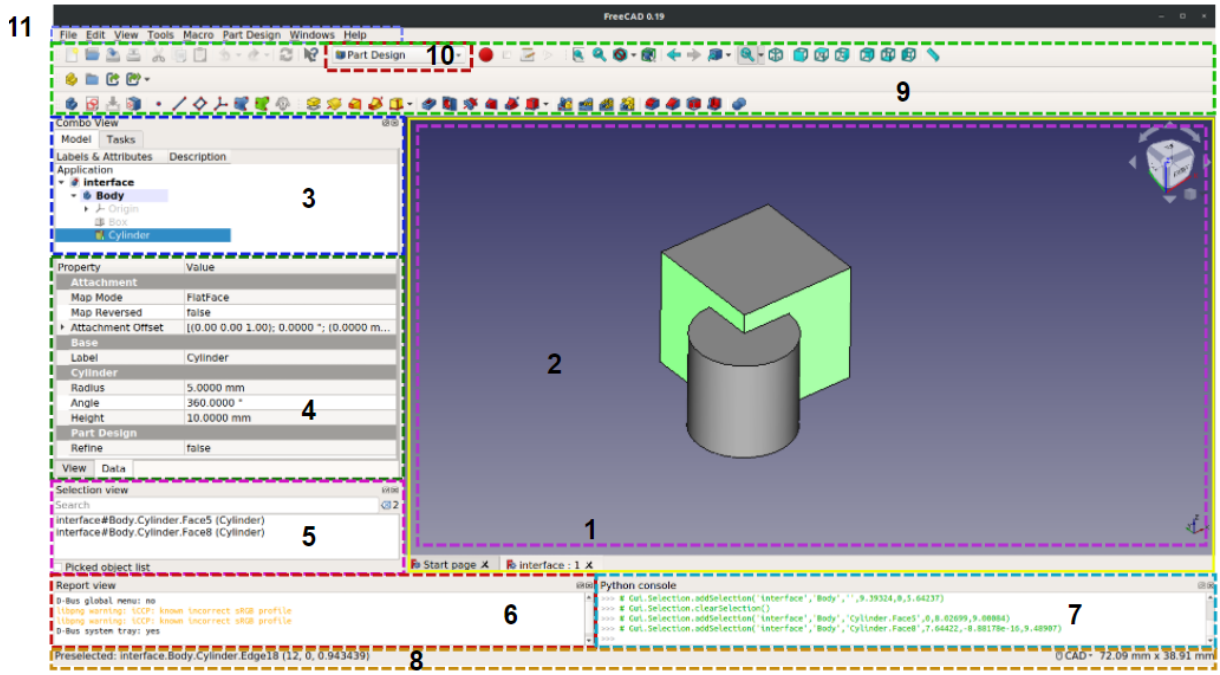


Figura 4.1: Interfaz del programa FreeCAD.

- Explicación de la interfaz:
  1. Área de visión: Puede contener diferentes ventanas con pestañas.
  2. Vista 3D: Muestra los objetos geométricos del documento.
  3. Vista árbol: Muestra los objetos en orden de construcción.
  4. Editor de propiedades: Permite ver y modificar las propiedades de los objetos seleccionados.
  5. Vista de selección: Indica los subelementos de los objetos (vértices,bordes,caras) seleccionados.
  6. Vista de informe: Muestra los mensajes, advertencias y errores.
  7. Consola Python: Muestra los comandos ejecutados y permite introducir el código Python.
  8. Barra de estado: Aparecen los tooltips y los mensajes.
  9. Área de la barra de herramientas: Zona donde está acoplada la barra de herramientas.
  10. Seleccionador de ambiente de trabajo.
  11. Menú estandar: Contiene las operaciones básicas del programa.

4.2. Robot IRB-120 ABB

El robot IRB 120 de ABB es un robot industrial de 6 ejes, con una carga útil de 3 kg que está diseñado principalmente para industrias de fabricación automática. También, es un robot con una estructura abierta que ofrece gran flexibilidad y adaptabilidad de comunicación con sistemas externos. Además, el robot admite software opcional, por lo que es compatible con aplicaciones como soldadura, comunicaciones de red o incluso funciones avanzadas como el procesamiento multitarea y el control de sensores.

En la siguiente figura se muestra el robot y el movimiento de los 6 ejes que componen su estructura [Figura 4.2].

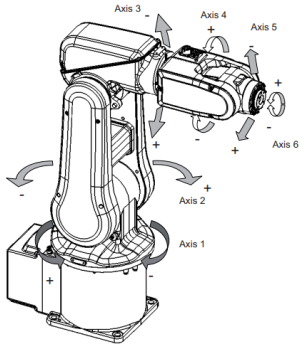


Figura 4.2: Ejes del robot IRB 120.

4.2.1. Especificaciones

En cuanto a las especificaciones generales del robot, tiene un peso de 25 Kg, un consumo de potencia de 0.24 kW, puede manejar una carga de hasta 3 kg y tiene un alcance de 0.58 metros. A continuación, sus dimensiones se pueden visualizar en la siguiente [Figura 4.3], para una información más detallada sobre las especificaciones del robot acceder a (ABB, 2010).

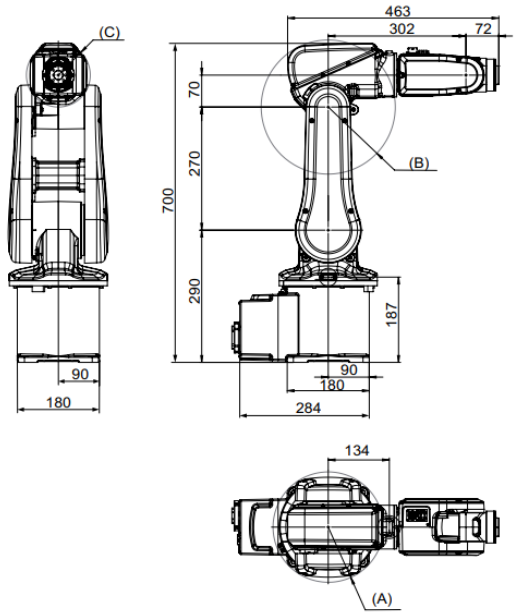


Figura 4.3: Dimensiones del robot IRB 120.

También se pueden visualizar las dimensiones de la brida para la herramienta del robot en la siguiente [Figura 4.4].

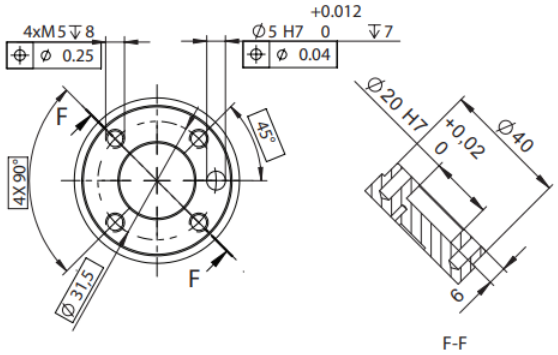
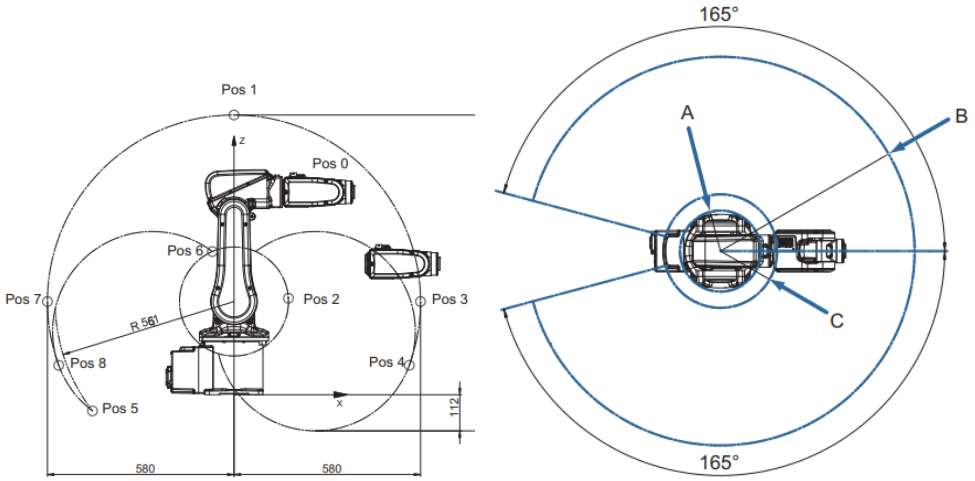


Figura 4.4: Dimensiones de la brida para herramientas del robot.

4.2.2. Área de trabajo

En las siguientes figuras se procede a mostrar el área de trabajo y el radio de giro que tiene el robot IRB 120. Las posiciones del extremo del brazo del robot están especificadas respecto del centro de la muñeca y las dimensiones es milímetros [Figura 4.5].



(a) Área de trabajo del robot. **(b)** Radio de giro del robot.

Figura 4.5: Límites del robot.

4.2.3. Conexiones del sistema neumático

Para la conexión neumática los conectores están situados en la base y la carcasa del brazo superior. Además, el robot tiene una manguera de aire comprimido integrada, como se puede observar en la siguiente [Figura 4.6], la base cuenta con 4 entradas que permiten la entrada de mangueras de diámetro exterior 4 milímetros y una presión de 5 bares (B).

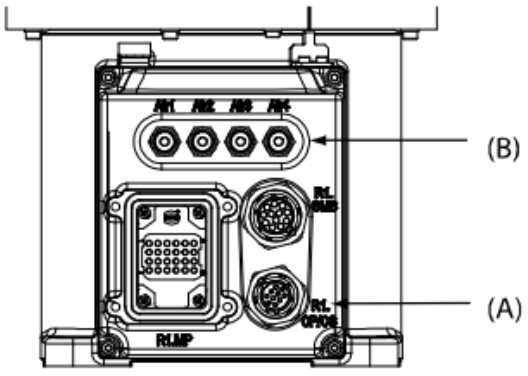


Figura 4.6: Entradas de las mangueras de aire comprimido.

4.2.4. Controlador del robot

El controlador IRC5 Compact es un controlador de robot de escritorio diseñado para segmentos como por ejemplo el mercado 3C (Cliente, compañía y competencia), la información sobre las especificaciones del controlador utilizado en el laboratorio se ha obtenido de (ABB, 2009). Además, es un controlador que ofrece flexibilidad, seguridad, modularidad, interfaces de aplicación, control de múltiples robots y compatibilidad con herramientas de PC [Figura 4.7].

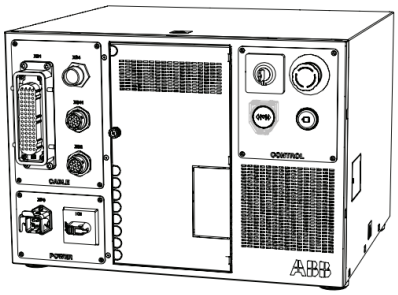
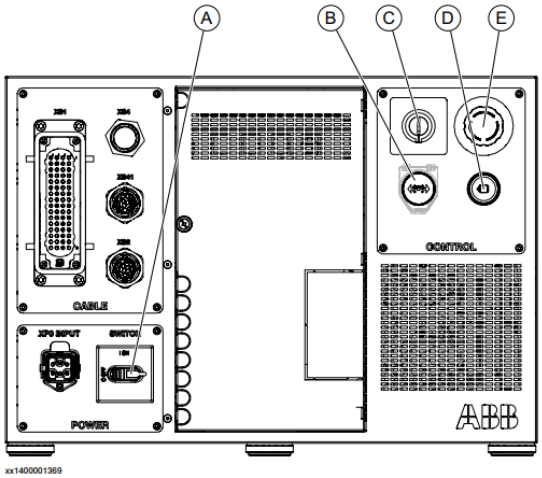


Figura 4.7: Controlador IRC5 Compact.

Los botones e interruptores del panel frontal se pueden visualizar en la siguiente [Figura 4.8].

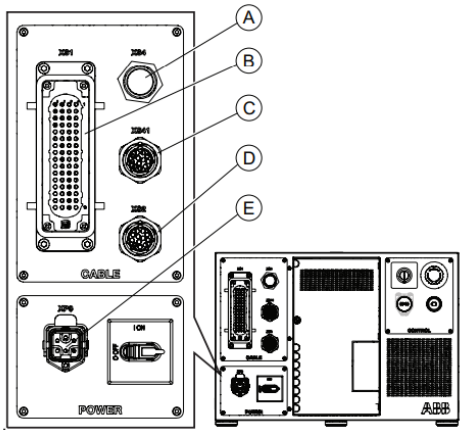


xx1400001369

A	Interruptor de alimentación principal
B	Botón de liberación de frenos (debajo de la cubierta) para el IRB 120. El IRC5 Compact utilizado con otros robots no tiene ningún botón de liberación de frenos, sino únicamente un conector de puente, dado que el robot es el que cuenta con un botón de liberación de frenos.
C	Selector de modo
D	Motores ON
E	Paro de emergencia

Figura 4.8: Controles del panel frontal.

A continuación, en la [Figura 4.9], se van a mostrar las conexiones del controlador.



(a) Panel de conexiones al controlador.

	Descripción
A	XS.4 Conexión de FlexPendant
B	XS.1 Conexión de alimentación del robot
C	XS.41 Conexión de tarjeta de medida serie de la tarjeta de medida serie
D	XS.2 Conexión de tarjeta de medida serie del robot
E	XP.0 Conexión eléctrica principal

(b) Especificaciones de las conexiones.

Figura 4.9: Conectores del controlador.

Cabe destacar que el controlador también contiene el software del sistema, el cual es el sistema operativo del robot. Este SO se llama RobotWare y proporciona todas las características necesarias para los aspectos fundamentales de la programación y el uso del robot. Específicamente, el controlador IRC5 Compact utiliza la versión 5.15 de RobotWare, para más información sobre el software de los controladores de ABB acceder a (ABB, 2004b).

4.2.5. FlexPendant

Todas las tareas de programación se pueden realizar utilizando el FlexPendant, el cual es una pantalla portátil que está conectada directamente al controlador. En la pantalla se muestra la información correspondiente al programa y al robot desde una interfaz intuitiva [Figura 4.10]. Además, desde (ABB, 2004a) se puede obtener toda la información respecto a la relación entre el controlador IRC5 y el FlexPendant.

(a) FlexPendant.

Pos	Descripción
A	Pantalla
B	Teclas programables
C	Botón de paro de emergencia
D	Joystick
E	teclas de ejecución de programas
F	Conexión de memoria portátil USB
G	Alojamiento de puntero

(b) Descripción del FlexPendant.

Figura 4.10: Especificaciones del FlexPendant.

En cuanto a las características de la pantalla, tiene un tamaño de 6.5 pulgadas, la introducción de datos por parte del usuario se realiza presionando los botones de la pantalla táctil. Además, permite tener abiertas varias pantallas a la vez para poder visualizar diferentes interfaces al mismo tiempo.

Cabe destacar el joystick tridimensional, que se utiliza para mover el robot manualmente durante los trabajos de programación. También el botón de paro de emergencia, que detiene el robot cuando se pulsa. Y finalmente, el botón de hombre muerto, el cual se encuentra en la parte de atrás del FlexPendant y obliga al usuario a mantenerlo pulsado a la mitad de su recorrido durante el control en modo manual para poder activar los motores y mover el robot.

4.3. RobotStudio

RobotStudio es un programa de ordenador que permite trabajar con datos del controlador IRC5, si se desea investigar en profundidad sobre el software acceder a (ABB, 2008). Además, se puede complementar con el FlexPendant para realizar la programación más eficiente. El FlexPendant tiene como objetivos controlar y ajustar el movimiento del robot de manera manual, y programar las posiciones de un entorno real. Por otro lado, RobotStudio tiene como finalidad manejar datos de configuración, gestionar programas y utilizar el acceso al robot de manera remota, esto es posible ya que RobotStudio puede actuar directamente sobre los datos activos del controlador.

Para poder conectarse al controlador RobotStudio tiene dos opciones. La primera es hacerlo localmente a través de la conexión PC de servicio. La segunda es conectarse desde la opción RobotWare PC Interface, a través de una conexión de red. Como RobotStudio posee un sistema de control maestro seguro, solamente puede tomar el control del sistema si el FlexPendant autoriza la petición de control.

A continuación, se va a describir la interfaz de usuario de RobotStudio junto con las herramientas principales que posee el programa [Figura 4.11].

Figura 4.11: Interfaz RobotStudio.

- Explicación de la interfaz:
    1. Archivo: Agrupa las opciones necesarias para crear una nueva estación, crear un sistema de robot, conectarse a un controlador y guardar o cargar una estación.
    2. Posición inicial: Agrupa los controles necesarios para construir estaciones, importar herramientas o bibliotecas, programar trayectorias y posicionar geometrías.
    3. Modelado: Contiene los controles necesarios para crear y editar componentes CAD.
    4. Simulación: Agrupa los controles necesarios para crear, configurar, controlar, monitorizar y grabar simulaciones.
    5. Controlador: Agrupa los controles necesarios para la sincronización, configuración y tareas asignadas al controlador virtual.
    6. RAPID: El editor de RAPID permite ver y editar los programas que estén cargados en la memoria de programas del controlador.
    7. Complementos: Contiene los controles de los PowerPac y de VSTA.

4.4. Sistema de visión In-Sight 7402C

In-Sight es un sistema de visión artificial compacto, independiente e integrable en redes para aplicaciones automatizadas de inspección, medición, identificación y control de robots en plantas de fabricación. El sistema de visión integrada de ABB ofrece un sistema de visión robusto e intuitivo enfocado a las aplicaciones de robótica guiada mediante visión. Este sistema proporciona un conjunto de software y hardware integrado con el controlador IRC5 y el entorno de programación de RobotStudio. Para el desarrollo de este sistema de visión ABB ha colaborado con la compañía Cognex, creando una serie de cámaras inteligentes que incorporan procesamiento integrado de imágenes junto con una interfaz de comunicación Ethernet, para más información sobre el sistema de visión integrada consultar (Cognex, 2008).

Figura 4.12: Cámara In-Sight 7402C ABB.

4.4.1. Especificaciones de la cámara In-Sight 7402C

En la siguiente [Figura 4.13] se reflejan las especificaciones generales del sistema de visión.

Especificaciones	In-Sight 7010/7020/7050/7200/ 7210/7230/7400/7410/7430	In-Sight 7010C/7200C/7400C	In-Sight 7402/7412/7432	In-Sight 7402C
Requisitos mínimos del firmware	In-Sight versión 4.7.1/4.7.3[1]	In-Sight versión 4.8.0	In-Sight versión 4.7.1/4.7.3[1]	In-Sight versión 4.8.0
Memoria de tarea/programa	Memoria flash no volátil de 512 MB; almacenamiento ilimitado a través de dispositivos de red remotos.			
Memoria de procesamiento de imágenes	256 MB de memoria SDRAM			
Tipo de sensor	CMOS de 1/1,8 pulg.			
Propiedades del sensor	Diagonal de 5,3 mm, píxeles de 5,3 x 5,3 µm cuadrados		Diagonal de 8,7 mm, píxeles de 5,3 x 5,3 µm cuadrados	
Resolución (píxeles)	800 x 600		1280 x 1024	
Velocidad del obturador electrónico	16 µs a 950 ms			
Adquisición	Restablecimiento rápido, barrido progresivo, integración de cuadros completos.			
Profundidad de bits	256 niveles de grises (8 bits/píxel)	Colores de 24 bits.	256 niveles de grises (8 bits/píxel)	Colores de 24 bits.
Ganancia y desplazamiento de la imagen	Controlado por software.			
Cuadros por segundo[2]	102 cuadros completos por segundo.	50 cuadros completos por segundo.	60 cuadros completos por segundo.	30 cuadros completos por segundo.
Tipo de lente	M12 o montura C.			
Variabilidad de alineación del sensor de imágenes[3]	±0,127 mm (0,005 pulg.), (tanto x como y) desde el eje de la montura C de la lente hasta el centro del generador de imágenes.			
Disparador	1 entrada de disparador de adquisición optoaislada. Comandos de software remoto a través de Ethernet y RS-232C.			
Entradas discretas	3 entradas de uso general cuando el dispositivo está conectado al cable de alimentación y de conexión de E/S. (Ocho entradas adicionales disponibles al usar el módulo opcional de E/S CIO-MICRO o CIO-MICRO-CC).			
Salidas discretas	4 salidas de alta velocidad cuando el dispositivo está conectado al cable de alimentación y de conexión de E/S. (Ocho salidas adicionales disponibles al usar el módulo opcional de E/S CIO-MICRO o CIO-MICRO-CC).			
Indicadores LED de estado	Enlace y actividad de red, alimentación y 2 configurables por el usuario.			

(a) Especificaciones 1.

Especificaciones	In-Sight 7010/7020/7050/7200/ 7210/7230/7400/7410/7430	In-Sight 7010C/7200C/7400C	In-Sight 7402/7412/7432	In-Sight 7402C
Lámpara anular LED interna	Rojo, verde, azul, blanco, IR (solo en configuración con lente M12).			
Comunicaciones de red	Puerto Ethernet, 10/100 BaseT, con MDI/MDIX automático. Protocolo TCP/IP IEEE 802.3. Ofrece direcciones IP DHCP (ajuste predeterminado de fábrica), estáticas y link-local.			
Comunicaciones serie	RS-232C: velocidades de transmisión de 4800 a 115200 baudios.			
Consumo	24 V CC ±10 %, 2,0 A.			
Material	Carcasa de aluminio.			
Acabado	Lacado.			
Montura	Cuatro orificios de montaje roscados M3 (1/4 - 20 y orificios de montaje M6 y de cabeza plana también disponibles en la abrazadera de montaje).			
Material del puerto de visión de la cubierta de la lente	Plástico de policarbonato transparente con un revestimiento resistente a la abrasión en la parte exterior.			
Dimensiones de la configuración con lente M12	55 mm (2,17 pulg.) x 84,8 mm (3,34 pulg.) x 55 mm (2,17 pulg.)			
Dimensiones de la configuración con lente de montura C	75 mm (2,95 pulg.) a 83 mm (3,27 pulg.) x 84,8 mm (3,34 pulg.) x 55 mm (2,17 pulg.) con la cubierta de la lente colocada.			
	42,7 mm (1,68 pulg.) x 84,8 mm (3,34 pulg.) x 55 mm (2,17 pulg.) con la cubierta de la lente sin colocar.			
Peso	220 g (7,8 onzas) con la cubierta y la lente M12 típica colocadas.			
Temperatura de funcionamiento	0°C a 45°C (32°F a 113°F)			
Temperatura de almacenamiento	-30°C a 80°C (-22°F a 176°F)			
Humedad	90 %, no condensante (en servicio y almacenado)			
Protección	IP67 (con la cubierta de la lente correctamente colocada).			
Impactos	Impactos de 80 G según IEC 60068-2-27.			
Vibración	10 G entre 10-500 Hz con la lente de 150 gramos, según IEC 60068-2-6.			
Cumplimiento de normas	CE, FCC, KCC, TÜV SÜD NRTL, RoHS			

(b) Especificaciones 2.

Figura 4.13: Especificaciones del sistema de visión.

A continuación se van a detallar los componentes del sistema de visión.

El cable Ethernet se utiliza para conectar el sistema de visión a otros dispositivos de red, en el caso de este proyecto se conectará directamente a los puertos del controlador IRC5 [Figura 4.15].

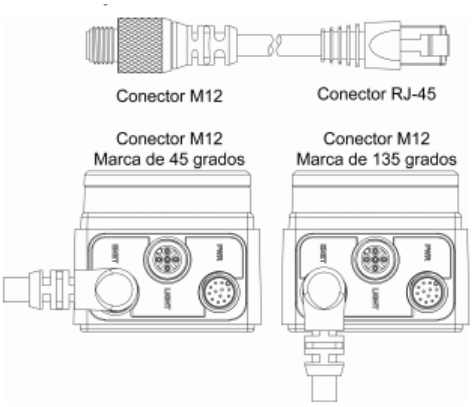


Figura 4.14: Cable y conexión Ethernet del sistema de visión.

El cable de alimentación y de conexión de E/S ofrece las conexiones a la fuente de alimentación externa, salidas de alta velocidad, comunicaciones RS-232 Serie y a la entrada del disipador de adquisición.

Figura 4.15: Cable y conexión de la alimentación y las E/S.

4.4.2. Dimensiones de la cámara In-Sight 7402C

Las dimensiones de la cámara con lente M12 se muestran en la siguiente [Figura 4.16]

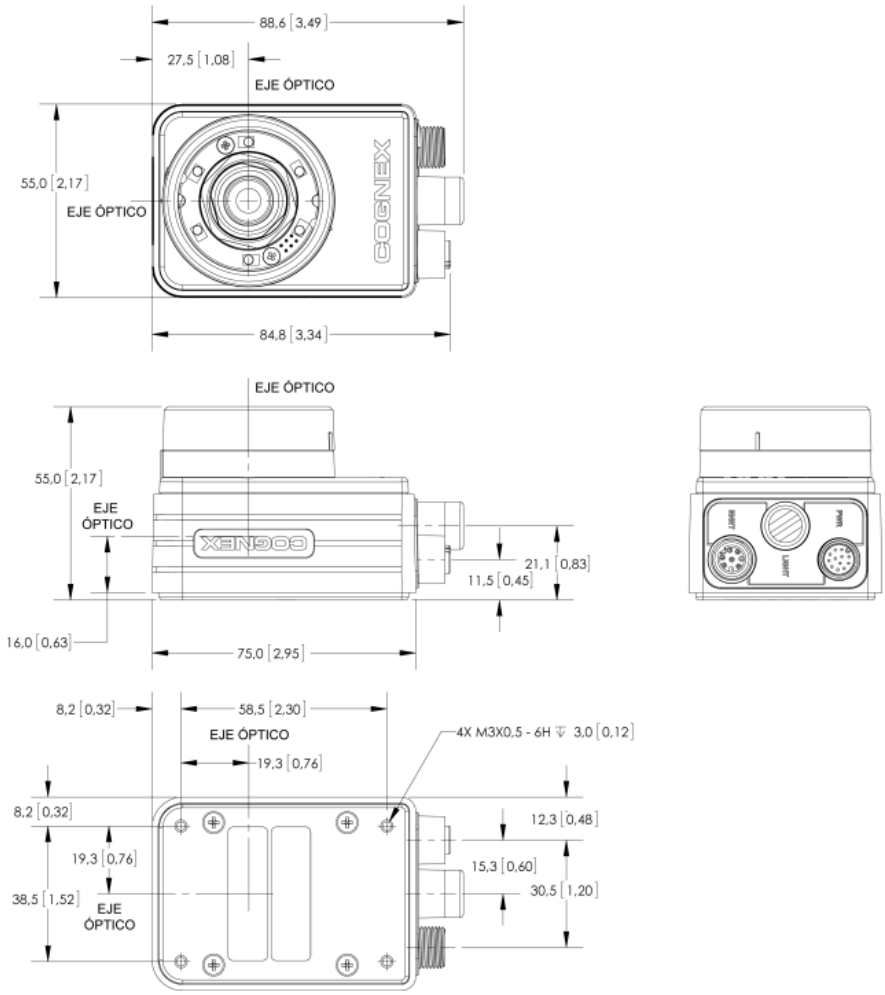
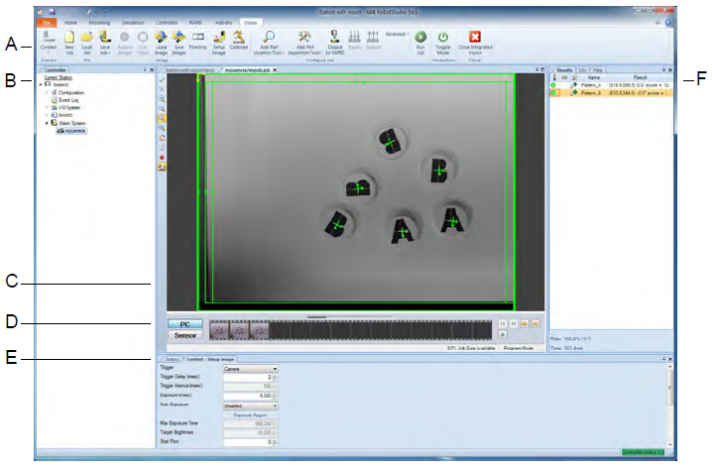


Figura 4.16: Dimensiones de la cámara con lente.

4.4.3. Software de la cámara In-Sight 7402C

RobotStudio está equipado con un entorno de programación de visión que expone toda la funcionalidad de Cognex EasyBuilder con herramientas robustas para localización, inspección e identificación de piezas. El lenguaje de programación RAPID ha sido ampliado con instrucciones específicas y seguimiento de errores para el manejo de las cámaras y el guiado por visión. La información respecto del software de la cámara se ha obtenido del manual de aplicación (ABB, 2018).

- La solución de software se basa en tres componentes:
    1. RobotStudio presenta los parámetros de visión y configuración del robot uno al lado del otro, proporcionando un entorno de programación intuitivo y sencillo, como se puede comprobar en la interfaz de usuario de RobotStudio [Figura 4.17].



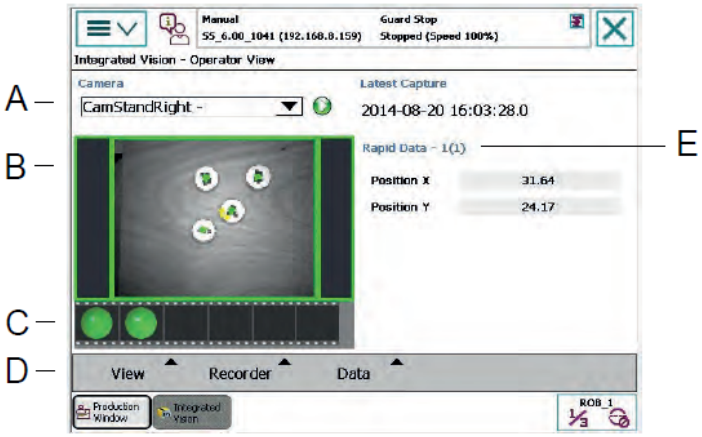
(a) Interfaz del sistema de visión integrada en RobotStudio.

	Repuestos	Descripción
A	Cinta	Muestra grupos de iconos organizados en una secuencia lógica de funciones.
B	Navegador Controlador	El nodo del sistema de visión muestra todas las cámaras de visión de la red.
C	Área de captura y configuración de imagen	Muestra una imagen adquirida por la cámara de visión, con guías de configuración para la localización y la inspección de piezas.
D	Secuencia de película	Se utiliza para grabar una secuencia de imágenes para su análisis posterior.
E	Ventana Contexto	Contiene las propiedades, la configuración y los eventos disponibles para los controles seleccionados.
F	Ventana de paleta	Están disponibles las siguientes pestañas: • Pestaña **Resultados**: muestra la configuración del trabajo de visión activo junto con una lista de todas las herramientas de localización e inspección utilizadas. • Pestaña **E/S**: muestra la configuración de E/S. • **Ayuda**: proporciona ayuda en línea.

(b) Especificaciones del sistema de visión integrada en RobotStudio.

Figura 4.17: Sistema de visión integrada en RobotStudio.

2. El FlexPendant está equipado con una interfaz de operador que permite la supervisión del sistema cuando se implementa durante la producción [Figura 4.18].



(a) Interfaz del sistema de visión integrada en el FlexPendant.

	Repuestos	Descripción
A	**Cámara**	Selecciona la cámara conectada y muestra el trabajo activo.
B	Área de imagen	Muestra una imagen adquirida por la cámara de visión.
C	Barra Recorder (Grabadora)	Se utiliza para grabar una secuencia de imágenes para su análisis posterior.
D	**Ver**	Se utiliza para cambiar la vista y mostrar la imagen, los resultados o ambos.
	Recorder (Grabadora)	Se utiliza para mostrar y congelar la barra Recorder (Grabadora) y para guardar imágenes.
	Data (Datos)	Se utiliza para configurar la vista de resultados.
E	Área de resultados	Se utiliza para mostrar los resultados de la cámara o datos de RAPID. Configurada por el usuario.

(b) Especificaciones del sistema de visión integrada en el FlexPendant.

Figura 4.18: Sistema de visión integrada en RobotStudio.

3. El controlador IRC5 permite creación de programas RAPID que aprovechan al máximo la capacidad del sistema de cámaras.

4.4.4. Límitaciones de la cámara In-Sight 7402C

- Para poder integrar el sistema de visión hay que tener en cuenta las siguientes limitaciones:
  1. Se requiere un FlexPendant del tipo SxTPU3 para ejecutar Integrated Visión.
  2. El complemento Integrated Vision no se ejecuta en la versión de 64 bits de RobotStudio.
  3. Las cámaras de Integrated Vision sólo están destinadas para su uso en la red de puerto de servicio del controlador de robot.
  4. Las cámaras deben configurarse con el complemento IntegratedVision de RobotStudio.
  5. Los programas de visión existentes realizados con Cognex EasyBuilder, deben ser modificados por el complemento Integrated Vision para que sean compatibles con el controlador IRC5.

5. Desarrollo

5.1. FreeCad

Para diseñar las piezas que se van a utilizar tanto en el entorno real como en el entorno de simulación, se ha utilizado el programa de diseño libre llamado FreeCad. Las piezas serán importadas a RobotStudio para poder utilizarlas dentro del espacio de trabajo de la estación virtual. Para trabajar en el entorno real se utilizará una impresora 3D que imprimirá todas las piezas diseñadas en FreeCad.

5.1.1. Diseño de herramienta de trabajo

Para el diseño de la herramienta de trabajo, el primer paso fue obtener las dimensiones de la herramienta real del laboratorio.

La herramienta está compuesta por tres figuras geométricas. La parte superior que corresponde a la ventosa, la parte intermedia que corresponde al núcleo de la herramienta y la parte inferior que corresponde a la base del elemento terminal y es la que está ensamblada con el extremo del robot.

- Dimensiones ventosa:
    1. Radio superior: 5 mm
    2. Radio inferior: 1 mm
    3. Altura: 5 mm
- Dimensiones núcleo:
    1. Radio: 5 mm
    2. Altura: 50 mm
- Dimensiones base:
    1. Radio: 25 mm
    2. Altura: 5 mm

El diseño en 3D en el programa FreeCad se muestra en la siguiente [Figura 5.1].

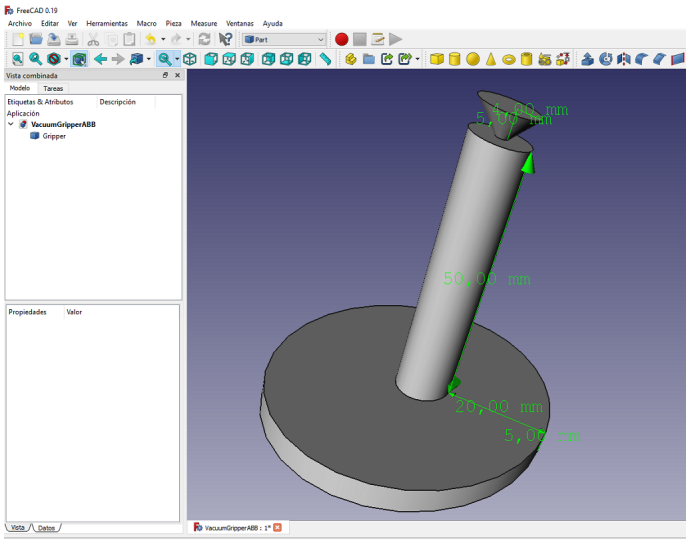


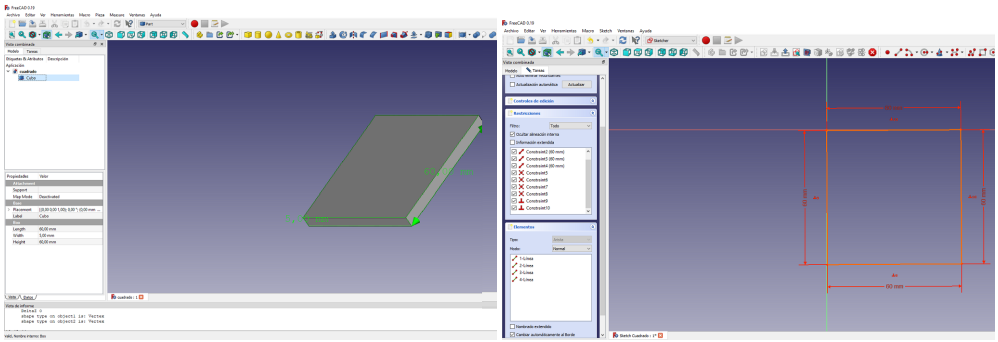
Figura 5.1: Diseño de la herramienta en FreeCad.

5.1.2. Diseño de objetos de trabajo

Al igual que la herramienta, tanto las piezas como el tablero se han diseñado en FreeCad. Las piezas son figuras geométricas simples y el tablero contiene unas zonas donde encajan perfectamente estas figuras.

Las dimensiones de las piezas son las siguientes:

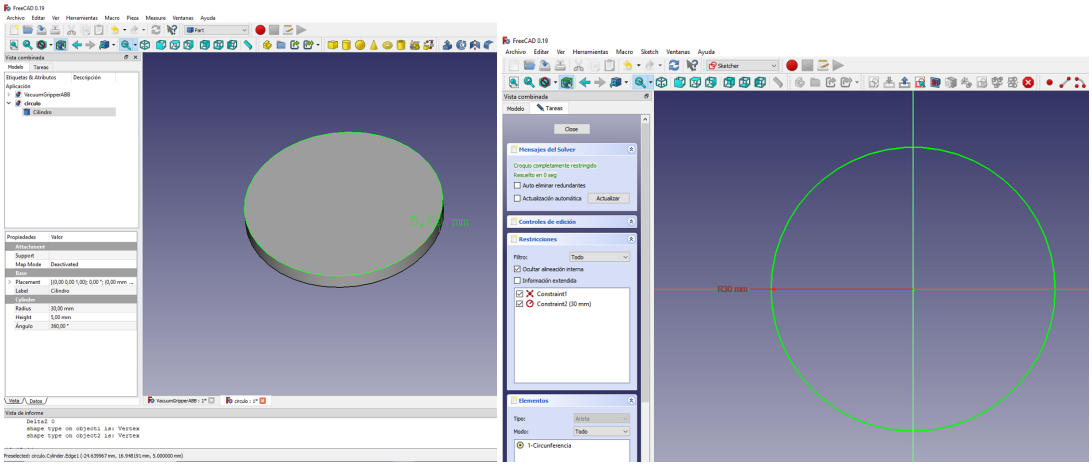
- Cuadrado [Figura 5.2]:
  1. Lado: 60 mm
  2. Ancho: 5 mm



(a) Diseño del Cuadrado. **(b)** Sketch del Cuadrado.

Figura 5.2: Cuadrado en FreeCad.

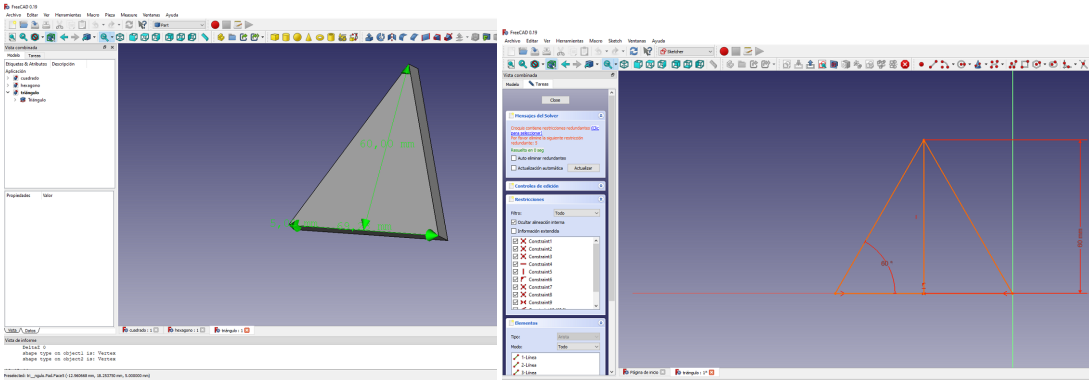
- Círculo [Figura 5.3]:
  1. Radio: 30 mm
  2. Altura: 5 mm



(a) Diseño del Círculo. **(b)** Sketch del Círculo.

Figura 5.3: Círculo en FreeCad.

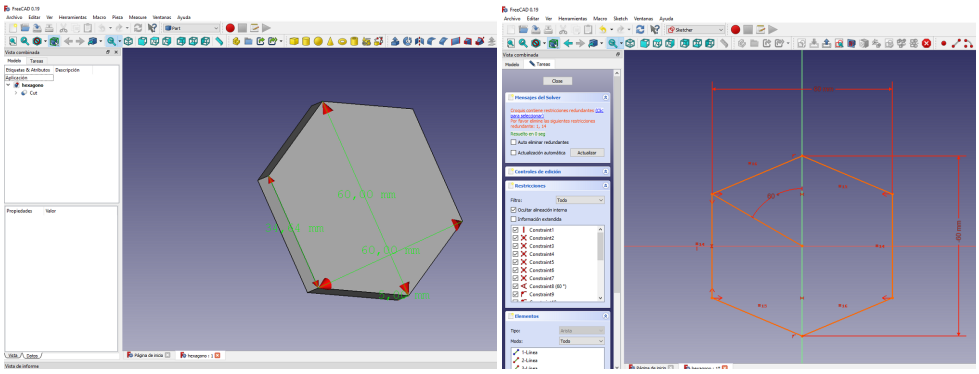
- Triángulo equilátero [Figura 5.4]:
  1. Altura: 60 mm
  2. Base: 69.23 mm
  3. Ancho: 5 mm



(a) Diseño del Triángulo. **(b)** Sketch del Triángulo.

Figura 5.4: Triángulo en FreeCad.

- Hexágono [Figura 5.5]:
  1. Altura: 60 mm
  2. Anchura: 5 mm
  3. Lado: 34,64



(a) Diseño del Hexágono. **(b)** Sketch del Hexágono.

Figura 5.5: Hexágono en FreeCad.

- Tablero [Figura 5.64a]:
  1. Lado: 200 mm
  2. Ancho: 5 mm
  3. Los huecos dónde deben encajar las figuras tienen una superficie 5 mm mayor para cualquier figura.

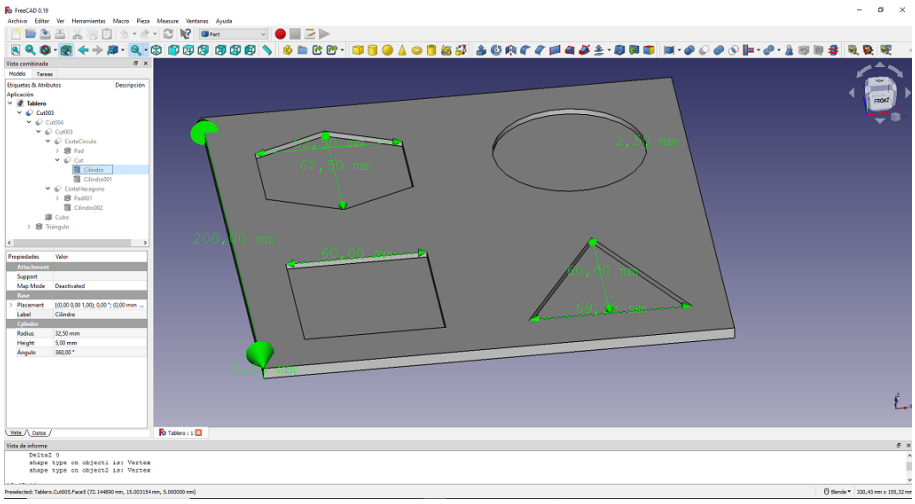


Figura 5.6: Diseño del tablero.

5.1.3. Exportar objetos

Para exportar los objetos que se han diseñado en FreeCad hay que tener en cuenta la extensión de la figura, ya que tienen que ser archivos compatibles con RobotStudio. Es por eso que todos los archivos creados han sido exportados con la extensión .step.

El primer paso para exportar una figura es dirigirse a la ventana "Archivo" y seleccionar "Exportar" [Figura 5.7].

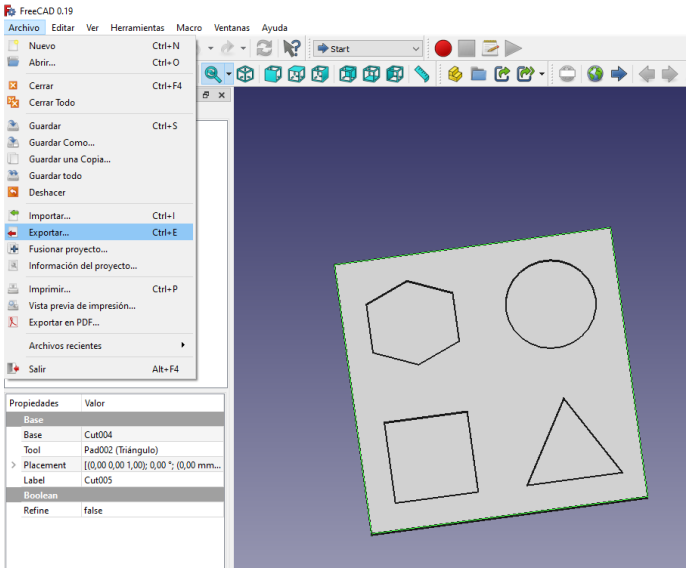


Figura 5.7: Herramienta exportar.

A continuación, aparecerá una ventana la cual permitirá guardar la figura en la ruta que especifiquemos, elegir la extensión y el nombre de la pieza [Figura 5.8].

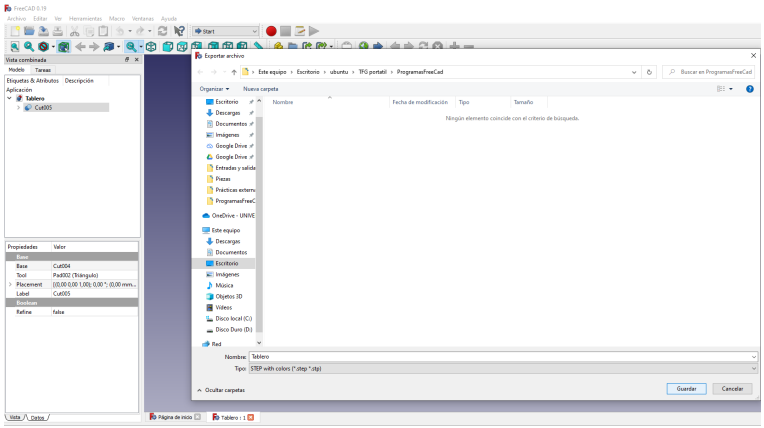


Figura 5.8: Exportar figura.

5.2. Estación robótica

Antes de proceder a la programación del comportamiento del robot, se debe tener clara la disposición del entorno de trabajo del robot. Por ello, en el entorno de simulación se van a realizar las configuraciones necesarias para recrear la estación de trabajo del entorno real. Además, se explicará como realizar la conexión desde un ordenador del laboratorio al controlador de un robot real.

5.2.1. Robot IRB-120 y controlador en entorno de simulación

En esta sección se van a explicar los pasos que hay que seguir para crear la estación robotizada en el entorno virtual, utilizando el mismo robot y controlador que se va a programar en el entorno real del laboratorio. Para ello se han tomado medidas de las dimensiones de la mesa de trabajo y de la base del robot del entorno real.

1. Robot: IRB-120 3Kg 580mm
2. Versión del controlador: RobotWare 6.10.02.00
3. Mesa de trabajo: 905 x 700 x 850 mm
4. Base del robot: 300 x 300 x 850 mm

5.2.1.1. Crear estación en entorno de simulación

El primer paso para crear la estación es abrir el software RobotStudio, seleccionar la ventana ”Nuevo”, y escoger la opción ” Solución con estación y controlador virtual” [Figura 5.9]. A continuación, seleccionar la pestaña ”Personalizar opciones”, escoger el nombre del robot, su modelo y el controlador que se ha especificado en la sección anterior [5.2.1].

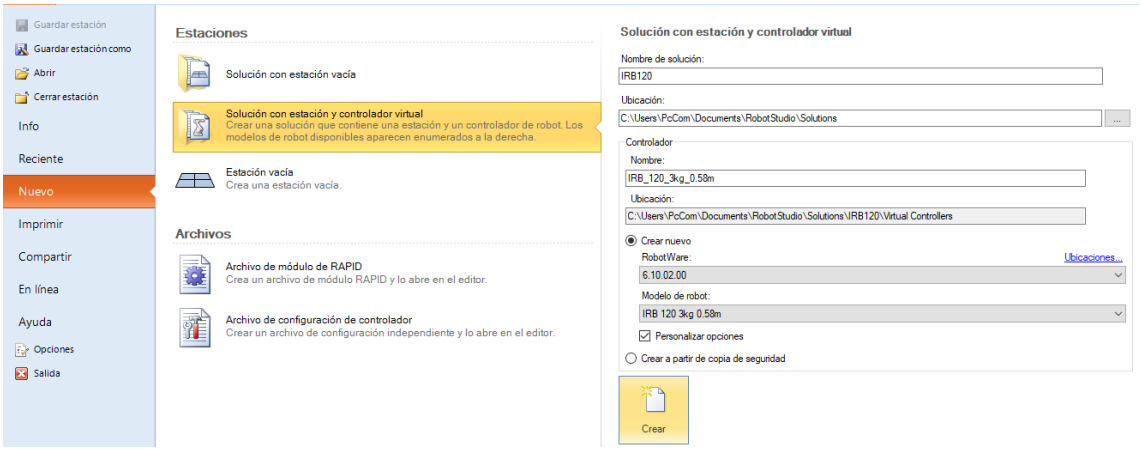


Figura 5.9: Crear estación con robot y controlador virtual.

Una vez creada la estación, emergerá una ventana que permitirá configurar el controlador del robot. Como en este proyecto se va a utilizar el sistema de visión integrada y la comunicación entradas-salidas digitales, es importante seleccionar la opción de visión y la comunicación maestro-esclavo [5.10]. Por último, aplicar los cambios realizados al controlador.

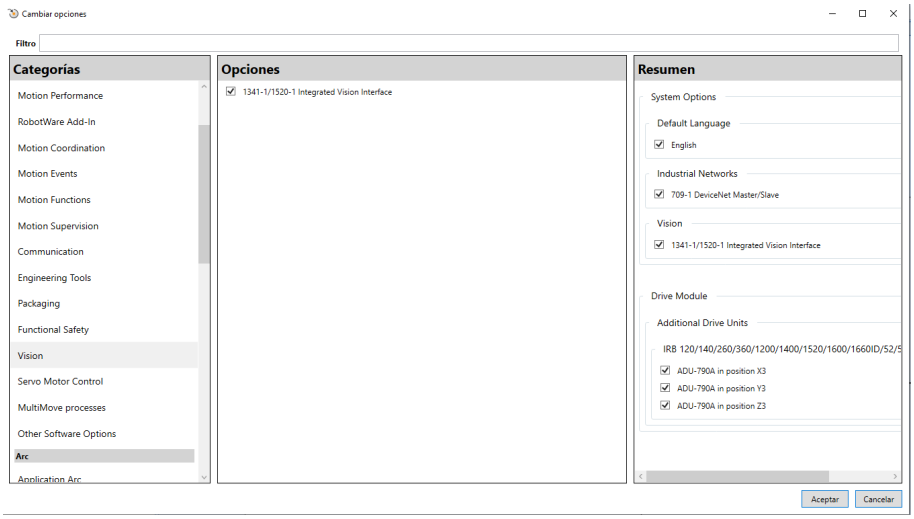


Figura 5.10: Configurar controlador.

Al finalizar la configuración del controlador aparecerá una ventana que nos permitirá escoger el robot que vamos a utilizar en el entorno de simulación [Figura 5.11].

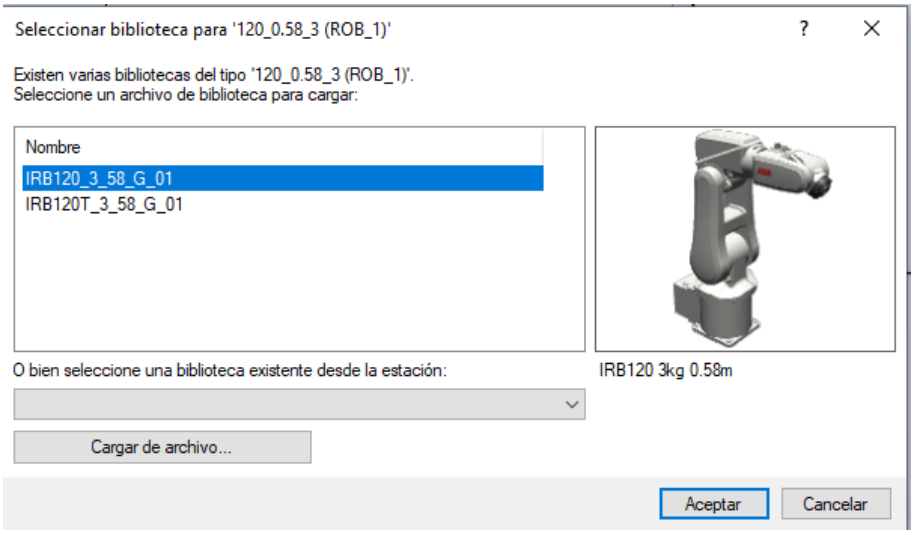


Figura 5.11: Escoger robot.

Finalmente, al terminar de configurar el controlador virtual y de importar el robot en la estación, la zona de trabajo quedaría de esta manera [Figura 5.12].

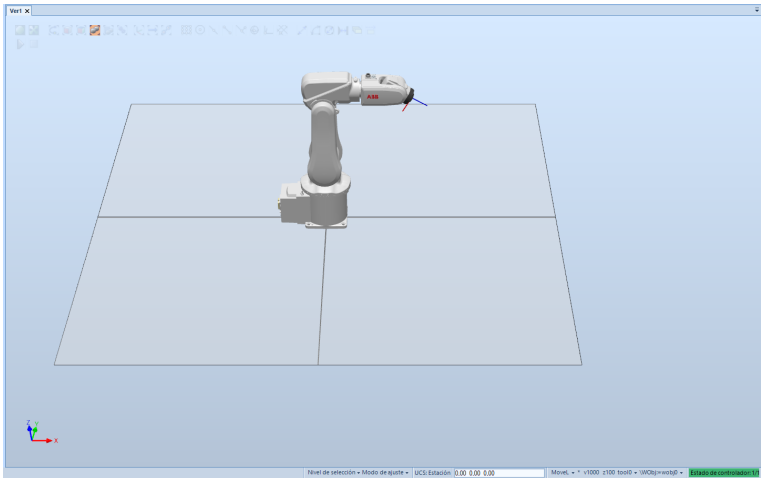


Figura 5.12: Estación con controlador virtual activo y robot importado.

5.2.1.2. Zona de trabajo

Una vez se ha configurado el controlador y se ha importado el robot, se procede a recrear la zona de trabajo del laboratorio en el entorno virtual.

El primer paso es crear tanto la mesa de trabajo como la base del robot con las medidas especificadas en la sección [5.2.1]. Para ello, dirigirse a la ventana "Modelado", seleccionar "Sólido" y escoger la opción "Tetraedro" [Figura 5.13].

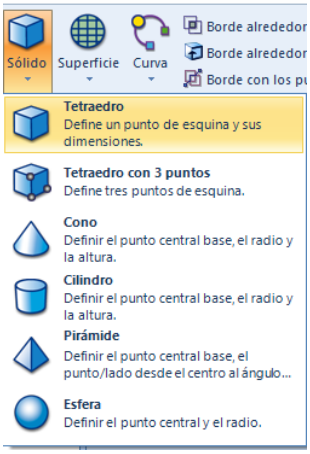
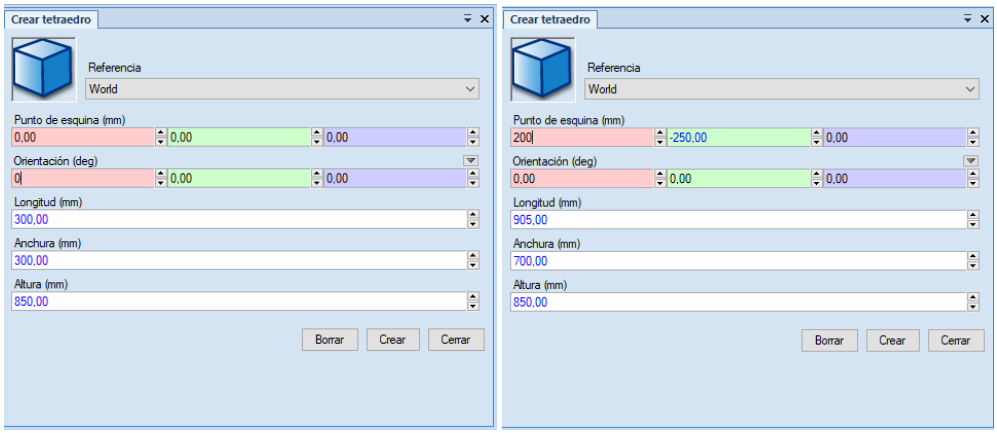


Figura 5.13: Herramienta crear sólido.

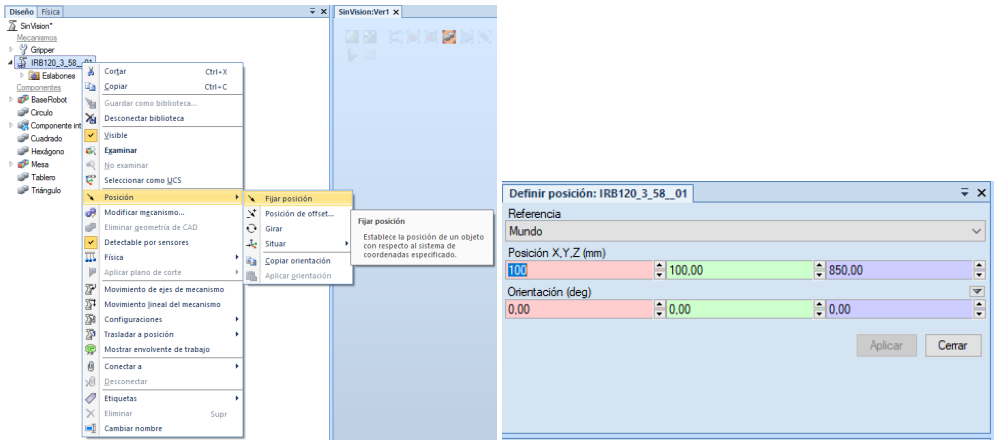
Aparecerá una ventana en la que se podrán definir las dimensiones y la posición del sólido que se desea crear [Figura 5.14].



(a) Dimensiones base del robot. **(b)** Dimensiones mesa de trabajo.

Figura 5.14: Crear sólidos.

A continuación, se debe posicionar el robot en la base creada. Dirigirse a la ventana "Diseño", seleccionar con el botón derecho del ratón el robot, utilizar la herramienta "Posición" y escoger la opción "Fijar posición" [Figura 5.15].



(a) Herramienta fijar posición. **(b)** Posición del robot en la estación.

Figura 5.15: Fijar posición del robot.

Finalmente, la estación con el controlador virtual configurado y la zona de trabajo replicando el laboratorio de robótica quedará como en la [Figura 5.16].

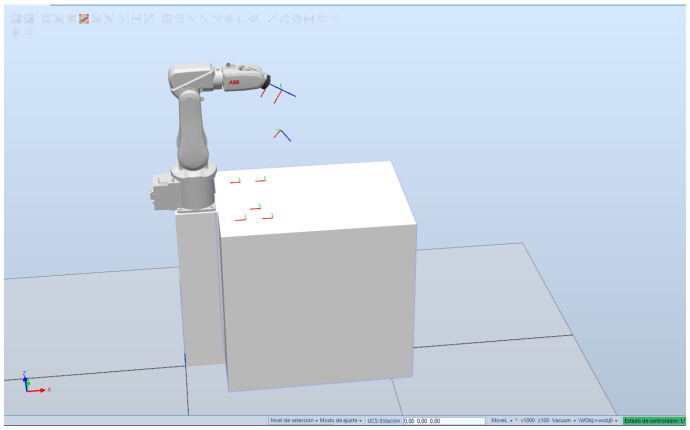


Figura 5.16: Estación en el entorno virtual.

5.2.2. Robot IRB-120 y controlador en entorno real

En esta sección se van a explicar los pasos que hay que seguir para establecer la conexión entre el controlador del robot y el ordenador con RobotStudio. Además, se mostrará la posición de los objetos y los dispositivos que conforman el entorno de trabajo de la estación.

5.2.2.1. Establecer conexión con el robot y el controlador real

Para poder programar el robot desde el ordenador es necesario establecer conexión con el controlador desde RobotStudio. El primer paso es dirigirse a la ventana "Archivo", seleccionar la opción "En línea" y escoger la alternativa "Añadir controlador" [Figura 5.17].

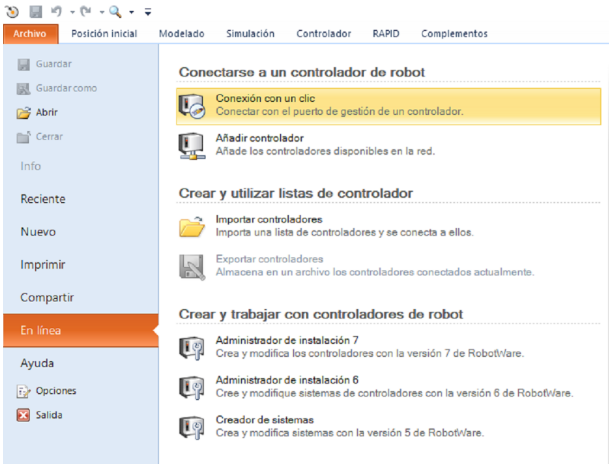


Figura 5.17: Añadir controlador en línea.

Aparecerá una ventana en la que se mostrarán todos los controladores que estén disponibles en la red local. Para abrir una estación con el controlador en línea solo hay que seleccionar el controlador activo [Figura 5.18].

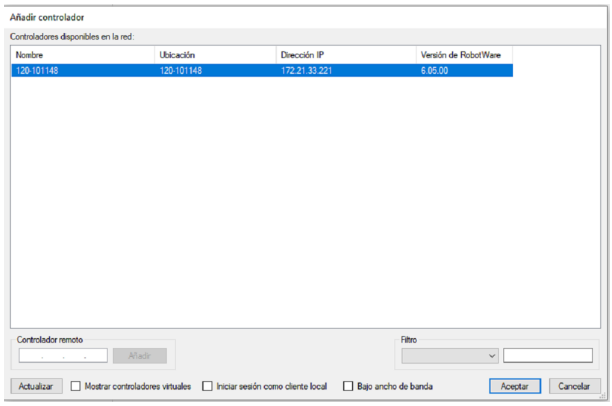


Figura 5.18: Establecer conexión con el controlador disponible en red.

5.2.2.2. Zona de trabajo

En cuanto a la zona de trabajo del robot real, esta se compone del robot, la cámara de visión, la mesa de trabajo y el FlexPendant.

La cámara está situada a 1,2 metros de altura de la mesa de trabajo [5.19].

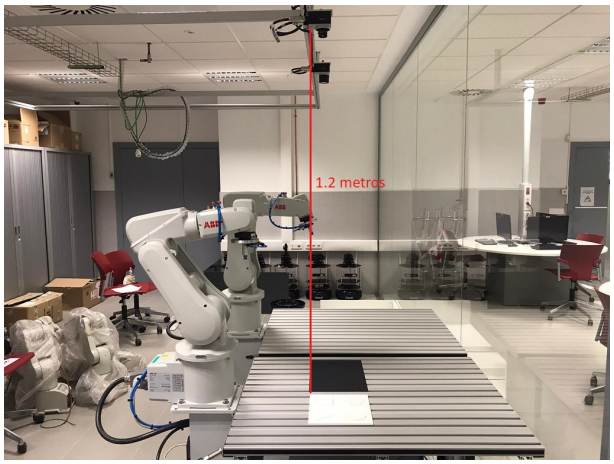


Figura 5.19: Distancia de la cámara a la mesa de trabajo.

5.3. Herramienta de trabajo

Una vez diseñada y creada la herramienta de trabajo, tenemos que cargar el archivo en RobotStudio y establecer el TCP del elemento terminal. El TCP es el punto central de la herramienta del robot, por lo que es el punto de trabajo que va a alcanzar las posiciones definidas en el espacio cartesiano.

5.3.1. Importar herramienta a RobotStudio

Para poder utilizar el objeto que se ha diseñado en FreeCad como una herramienta en RobotStudio, primero se debe importar el archivo desde la ventana "Modelado" utilizando la opción "Importar geometría" y seleccionando "Buscar geometría" [Figura 5.54].

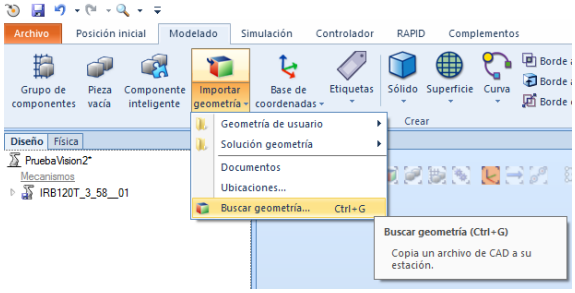


Figura 5.20: Importar geometría.

A continuación, hay que seleccionar el archivo que contenga la geometría de la herramienta de trabajo. Para ello es importante elegir el archivo que tenga extensión .step y seleccionar la opción "Convertir geometría de CAD en pieza individual" para que RobotStudio reconozca la geometría y podamos crear la herramienta [Figura 5.21].

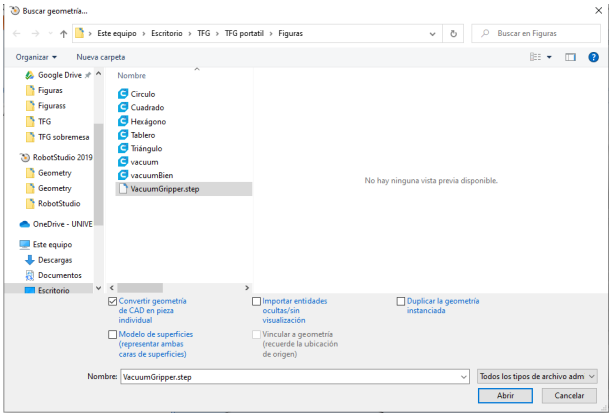
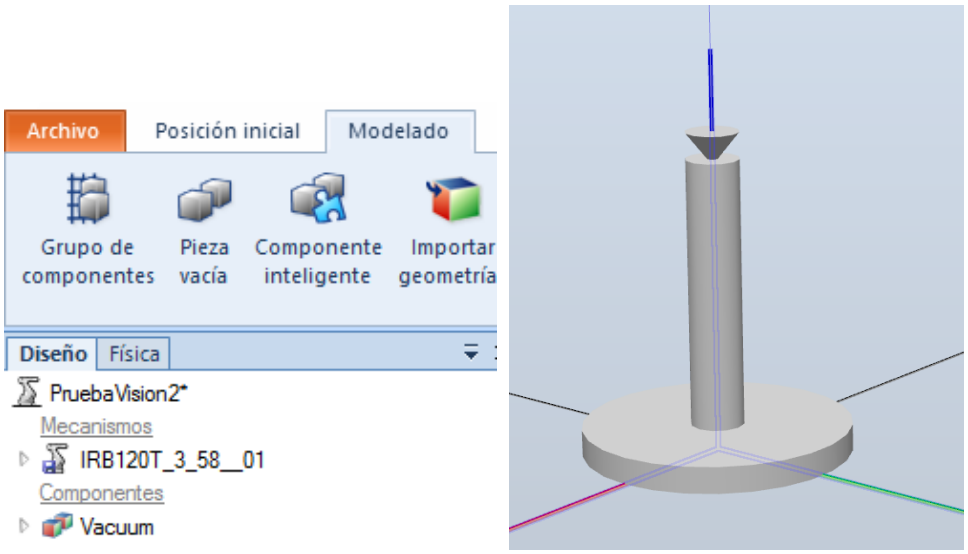


Figura 5.21: Importar herramienta.

Como se puede comprobar en las [Figuras 5.22], la herramienta se ha importado correctamente, pero es solamente un sólido, por lo que hay que hacer que el programa reconozca esta geometría como una herramienta y establecer el TCP en el extremo de la ventosa.



(a) Herramienta como geometría. **(b)** Herramienta sin TCP.

Figura 5.22: Componente geométrico de la herramienta importada.

5.3.2. TCP

Una vez la geometría de la herramienta está dentro del espacio de trabajo del robot, se procede a crear el elemento terminal. Para ello, desde la ventana "Modelado" se selecciona la opción "Crear herramienta" [Figura 5.23].

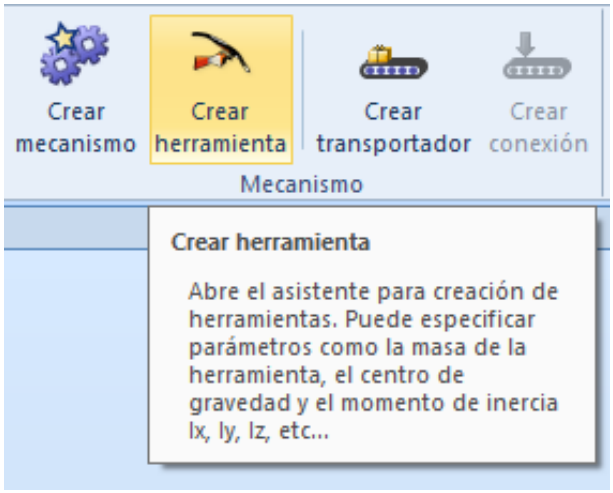


Figura 5.23: Crear herramienta.

Aparecerá la ventana "Información de herramientas" [Figura 5.24], en la que primeramente se establece el nombre de la herramienta. A continuación, se selecciona la opción "Usar existente" y se escoge la geometría que se ha importando anteriormente "Vacuum". Además, se modifica la masa de la herramienta a 0.4 kg (una aproximación del peso de la herramienta real).

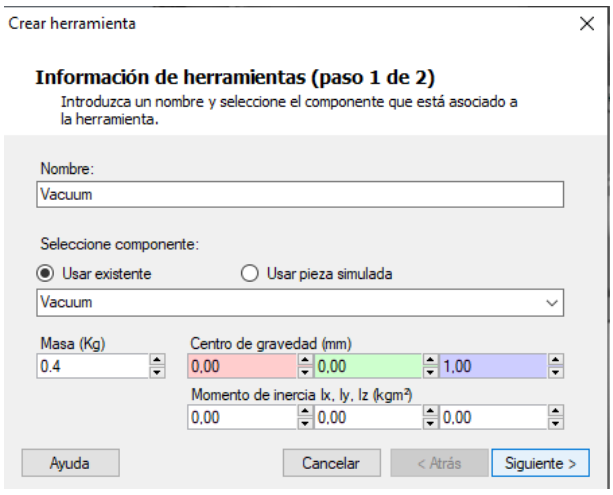


Figura 5.24: Información de herramientas.

El segundo paso es establecer el sistema de referencia del TCP, para ello se selecciona la posición teniendo en cuenta como referencia el WorldObject, que es el sistema de referencia del entorno de simulación. Como se puede apreciar en la [Figura 5.25], el TCP está en la posición [0,0,60]mm que es la correspondiente al extremo de la ventosa.

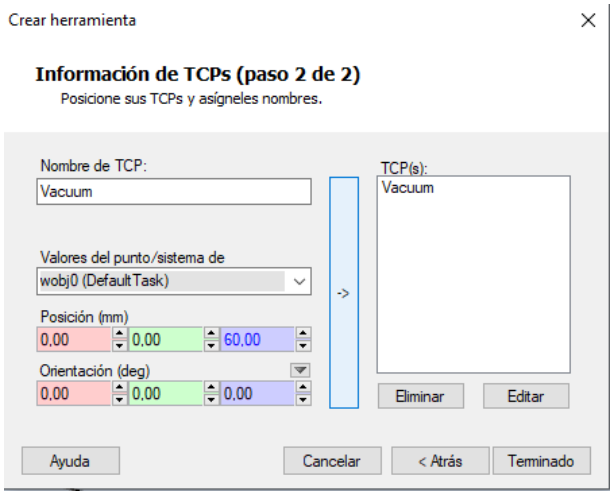
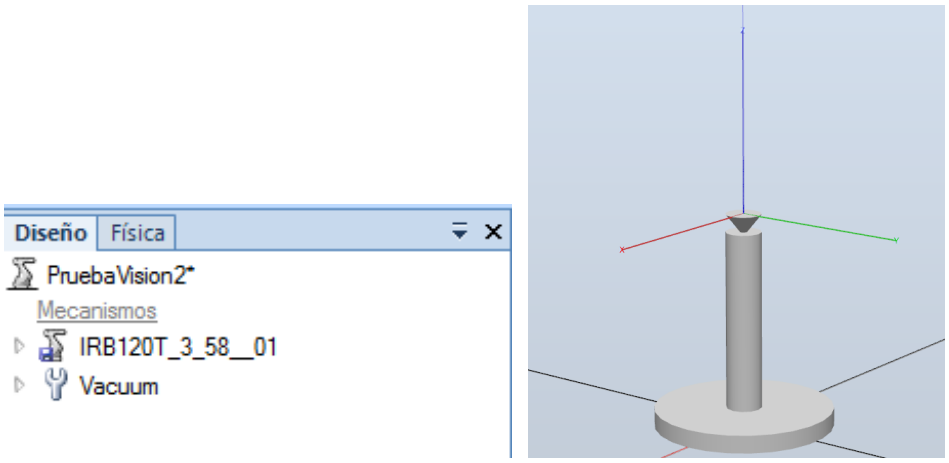


Figura 5.25: Información de TCPs.

Una vez se ha terminado de crear la herramienta, la ventosa debería aparecer como un mecanismo en la ventana de diseño. Además, el TCP debería de aparecer en el extremo de la herramienta [Figura 5.26].



(a) Herramienta como mecanismo. **(b)** Herramienta con TCP.

Figura 5.26: Herramienta creada correctamente.

Por último, para ensamblar la herramienta creada en el extremo del robot, desde la ventana "Diseño", se selecciona la herramienta y se arrastra encima del robot. Aparecerá el siguiente mensaje [Figura 5.27]. De esta manera, se actualizará la posición de la herramienta con la del extremo del robot quedando como resultado la [Figura 5.28].

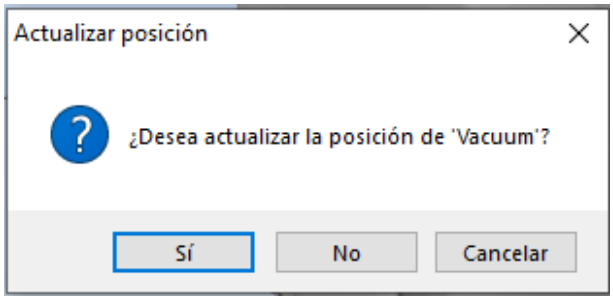


Figura 5.27: Actualizar posición de la herramienta.

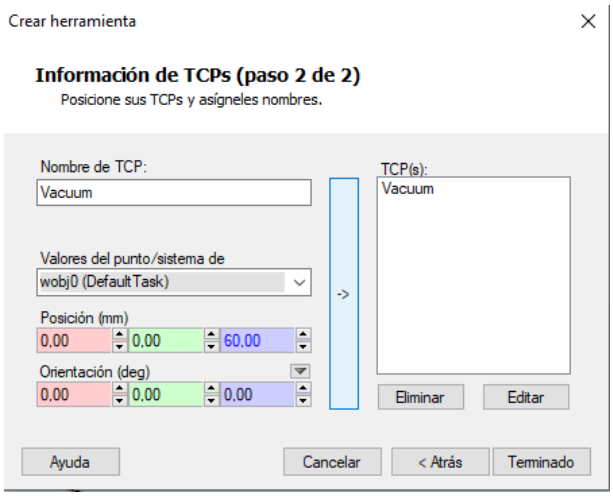


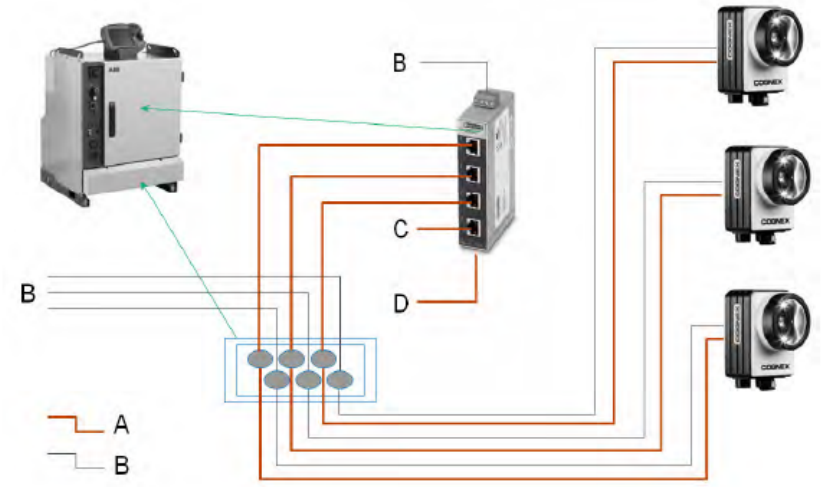
Figura 5.28: Robot con la herramienta conectada.

5.4. Conexiones

En esta sección se van a explicar las conexiones realizadas para poder utilizar e instalar tanto la neumática como la cámara de visión integrada en la aplicación del entorno real.

5.4.1. Cámara de visión

Para conectar la cámara de visión hay que realizar la conexión de los componentes y cables siguiendo el esquema mostrado en la siguiente [Figura 5.29].



xx1200000992

A	Ethernet
B	Alimentación a 24 V CC desde la fuente de alimentación del usuario al switch y a las cámaras
C	Ethernet entre el switch y el puerto de servicio del armario del controlador (interior)
D	Ethernet entre el switch y el puerto de servicio del ordenador principal

Figura 5.29: Esquema de conexión de la cámara al controlador.

Para realizar la instalación de la cámara primero hay que asegurarse de que el interruptor eléctrico del controlador esté apagado. A continuación, conectar un cable de Ethernet entre el puerto de servicio del armario del controlador y uno de los cuatro conectores de Ethernet del switch. Después, conectar el cable de Ethernet de la cámara a cualquier switch disponible del controlador. Finalmente, conectar los cables de alimentación de 24 V CC de la cámara a la fuente de alimentación de 24 V CC del controlador.

5.4.2. Sistema neumático

Primero hay que ver las conexiones neumáticas que tiene el robot [Figura 5.30].

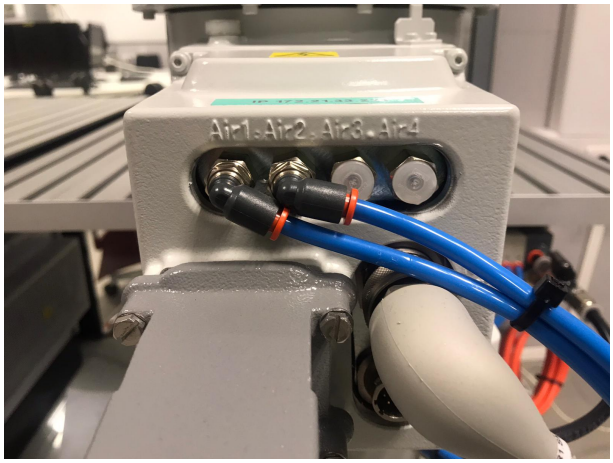


Figura 5.30: Conexiones neumáticas al robot.

Para el control de la neumática se van a utilizar las salidas digitales de la tarjeta DSQ652 integrada en el controlador del robot. Como las conexiones de aire son Air1 y Air2, se van a conectar a las salidas digitales de la tarjeta DO1 y DO2 [Figura 5.31]. Dónde la salida DO1 en el programa de RobotStudio será la señal "Soplado" encargada de crear el vacio en la ventosa y la salida DO2 será la señal "Vacio" encargada de expulsar aire, estas señales se pueden comprobar en la [Sección 5.7.2.2].

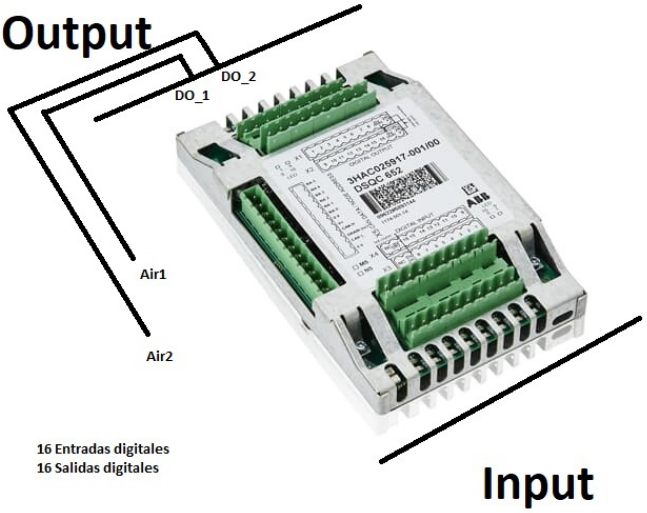


Figura 5.31: Conexiones a la tarjeta de control DSQ652.

5.5. Sistema de visión In-Sight 7402C

Para poder integrar el sistema de visión en el robot, primeramente se deberán haber realizado las conexiones físicas de la cámara con el controlador [5.4.1]. Además, para poder utilizar la cámara para detectar objetos en el espacio de trabajo de visión, es necesario sincronizar la cámara con nuestra estación de RobotStudio y realizar un buen calibrado de esta.

5.5.1. Sincronizar cámara con RobotStudio

El primer paso para poder integrar el sistema de visión una vez la cámara está conectada físicamente al controlador, es configurar la IP de la cámara y proporcionarle un nombre. El nombre de la cámara es el identificador que lo diferenciará en todas las partes del sistema, como en RobotStudio, o en los programas de RAPID. De manera predeterminada, la dirección IP de la cámara se asigna de forma automática por el controlador mediante DHCP, pero en este caso se establecerá una dirección IP estática. Esto es debido al hecho de que la dirección IP de la cámara no corresponde a la misma subred que la del controlador y el PC.

Para que la cámara se conecte a la misma subred que el controlador, primero se debe configurar el ordenador conectado al controlador para que ambos estén conectados a la misma subred [Figura 5.32]. Una vez los dos dispositivos comparten subred, desde RobotStudio se le asigna una dirección IP fija a la cámara.

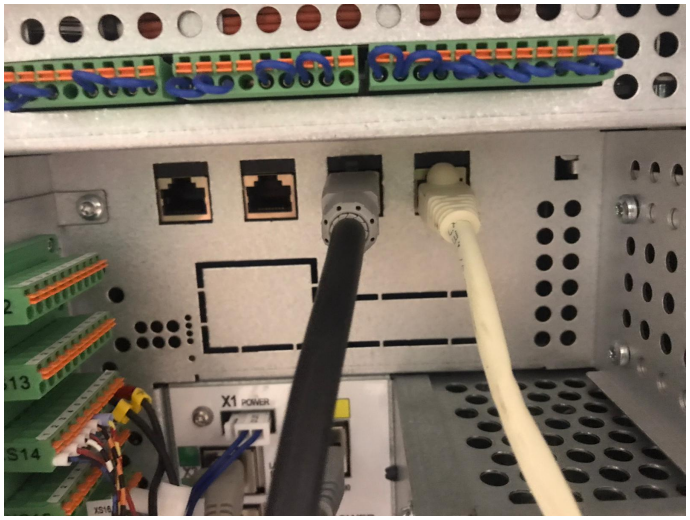


Figura 5.32: Conexión de la cámara al controlador(Cable izquierdo) y conexión del controlador a un PC(Cable derecho).

Una vez realizadas estas conexiones, nos dirigimos a "Conexiones de red" del PC conectado al controlador [Figura 5.33].

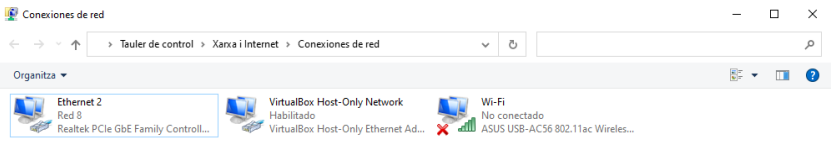
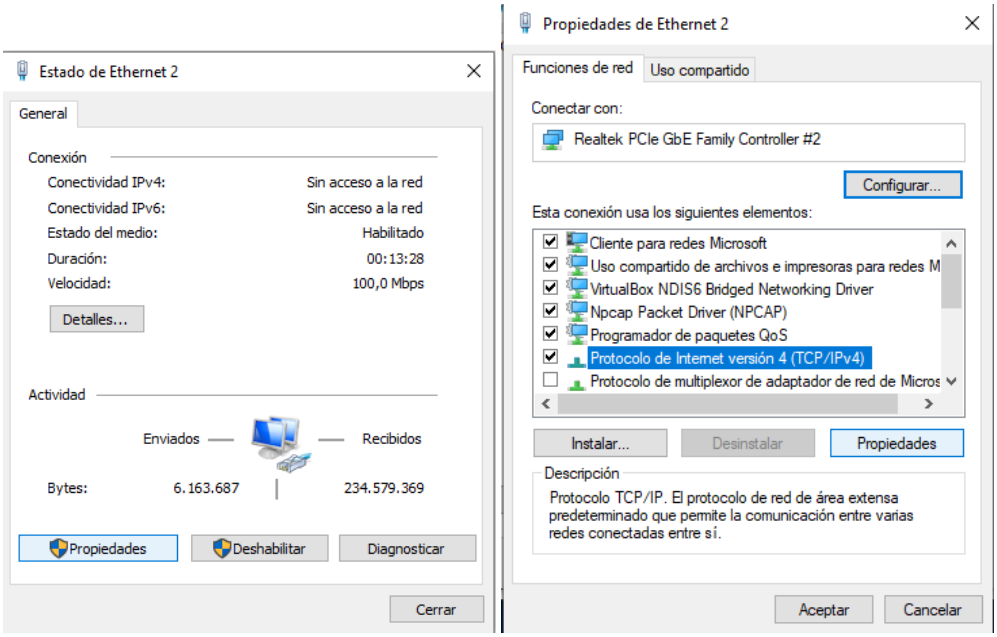


Figura 5.33: Conexiones de red.

Desde "Conexiones de red", en nuestro caso seleccionamos "Ethernet 2", y accedemos a la opción propiedades [Figura 5.34a]. Emergerá la ventana de propiedades de la red seleccionada, como queremos utilizar una IP fija iremos a las propiedades del protocolo IPv4 [Figura 5.34b].



(a) Estado de red. **(b)** Propiedades de red.

Figura 5.34: Configuración de red.

Una vez en la ventana de propiedades del protocolo Ipv4, configurar la dirección IP dentro del rango 192.168.125.X, para que tanto el controlador como el PC se encuentren en la misma subred [Figura 5.35].

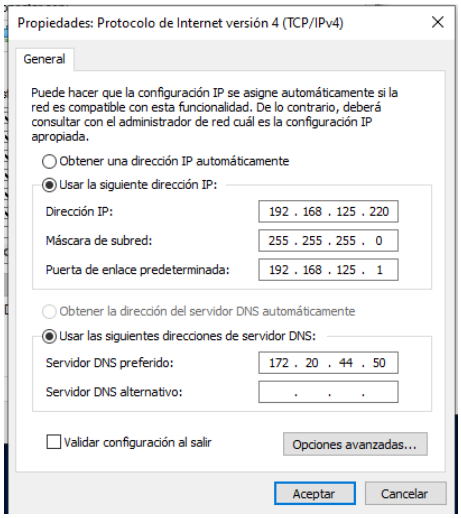
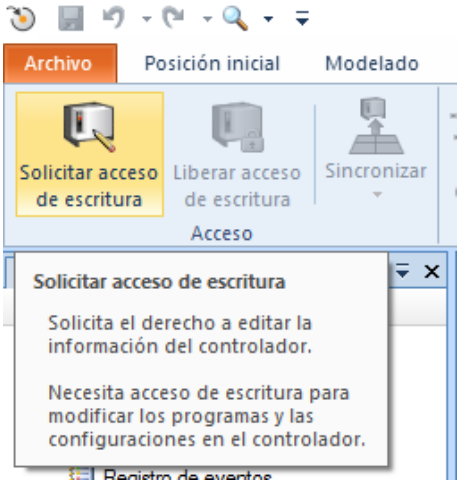
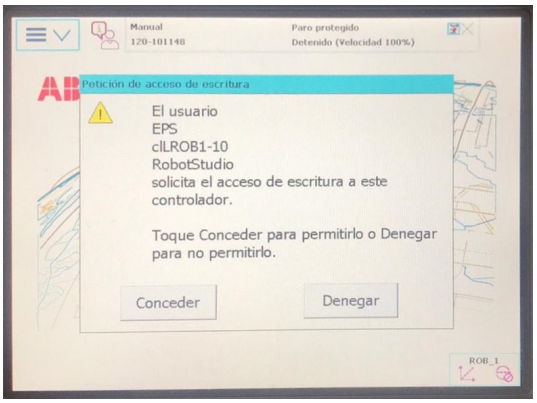


Figura 5.35: Modificar IP del PC.

A continuación, abrir una estación en RobotStudio que esté conectada al robot real, dirigirse a la ventana "Controlador" y seleccionar "Solicitar acceso de escritura" [Figura 5.36a]. En el FlexPendant aparecerá un mensaje, al aceptarlo se concederá el acceso a la programación y configuración del robot desde el PC que estamos utilizando [Figura 5.36b].



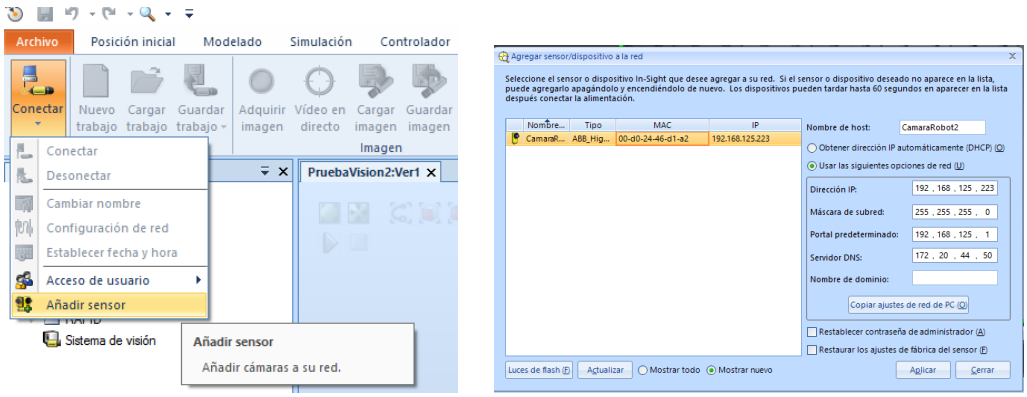
(a) Solicitar acceso de escritura.



(b) Aceptar acceso de escritura.

Figura 5.36: Acceso de escritura.

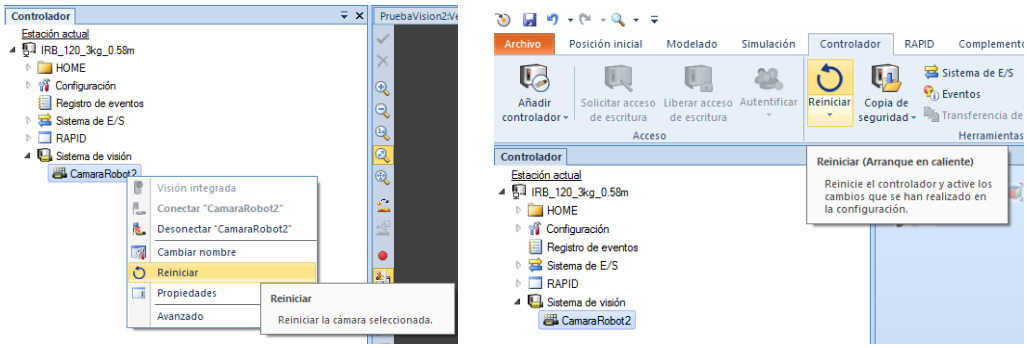
Seguidamente, dirigirse a la ventana de ”Visión” y seleccionar la opción ”Conectar” y elegir ”Añadir sensor” [Figura 5.37a]. Aparecerá una ventana que permitirá añadir la cámara a la red [Figura 5.37b]. Seleccionar la cámara, y copiar los ajustes de red del PC, cabe destacar que es importante elegir el nombre de la cámara (CamaraRobot2 en nuestro caso).



(a) Añadir cámara a la estación. **(b)** Establecer IP fija a la cámara.

Figura 5.37: Sincronizar cámara con RobotStudio.

Una vez se terminada la configuración, aplicar los cambios realizados, reiniciar la cámara y reiniciar el controlador [Figura 5.38].



(a) Reiniciar cámara. **(b)** Reiniciar controlador.

Figura 5.38: Aplicar cambios realizados a la cámara.

Finalmente, dirigirse a la ventana de "Visión" y seleccionar "Conectar CámaraRobot2" [Figura 5.39].

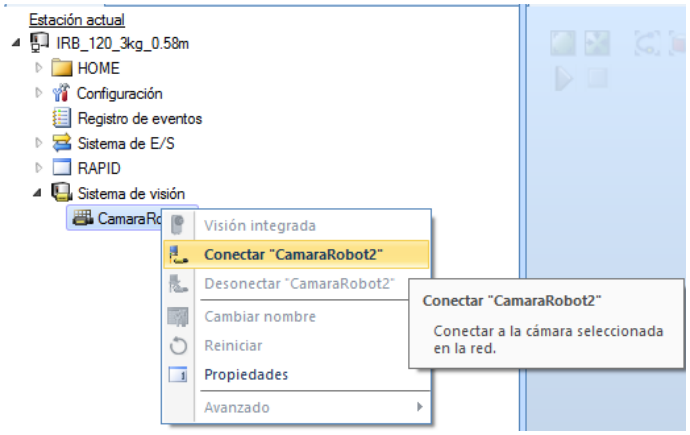


Figura 5.39: Conectar cámara.

Automáticamente, la cámara se conectará y podremos utilizarla para continuar con la programación del sistema robótico [Figura 5.40].

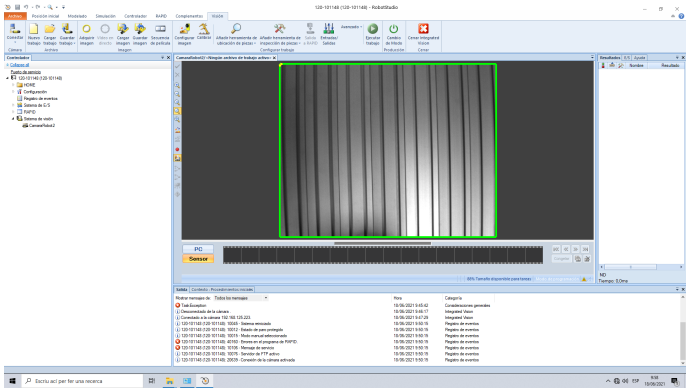


Figura 5.40: Cámara conectada correctamente.

5.5.2. Calibrar cámara

Las imágenes están compuestas de píxeles, por lo que para conseguir el resultado en mm es necesario calibrar la cámara. En este proyecto, para realizar la calibración se va a utilizar una función específica desarrollada por el entorno de simulación de ABB. La función llamada "Calibrar" se encuentra en la ventana de "Visión" y se utiliza para calibrar las imágenes obtenidas por la cámara a unidades del mundo real [Figura 5.41].

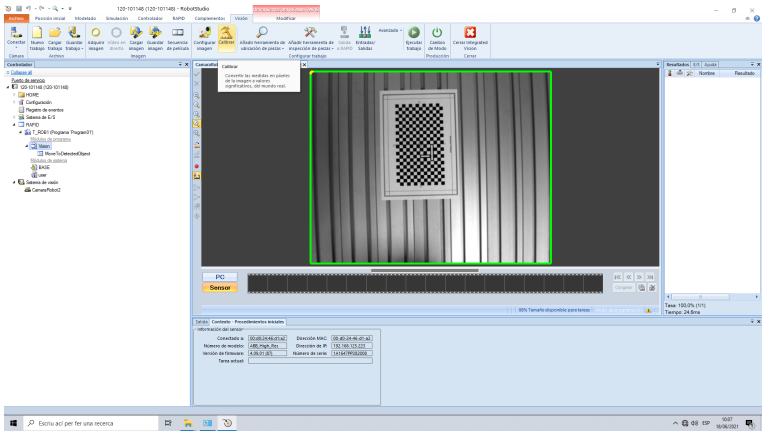


Figura 5.41: Función calibrar.

Una vez seleccionada la función, utilizamos la herramienta "Adquirir imagen" para obtener la foto de la vista de la cámara. Como se puede apreciar en la [Figura 5.42], el tablero está dispuesto en una orientación en la que los ejes X e Y tienen una orientación clásica. Además, el tablero está en una posición centrada del campo de visión de la cámara para que no hayan problemas en la calibración.

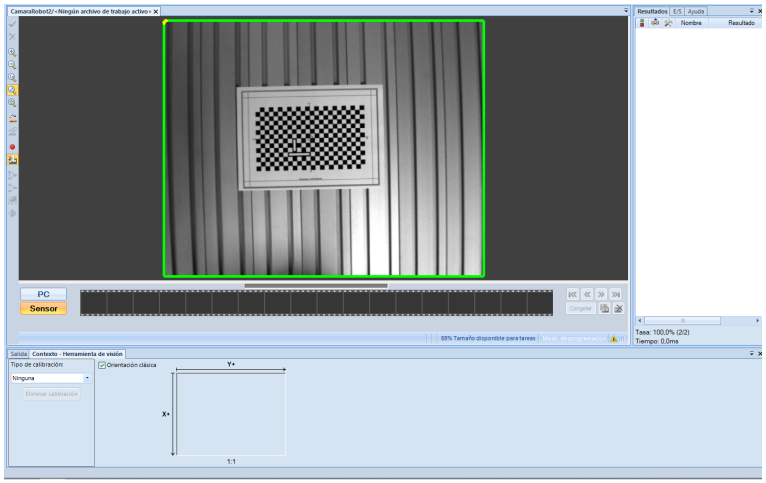


Figura 5.42: Posicionar tablero.

El siguiente paso es seleccionar el tipo de calibración, este depende del tipo de tablero que se utilice para este fin. En este proyecto se ha utilizado un tablero de tipo grid con un espaciado de 10mm entre celdas [Figura 5.43].

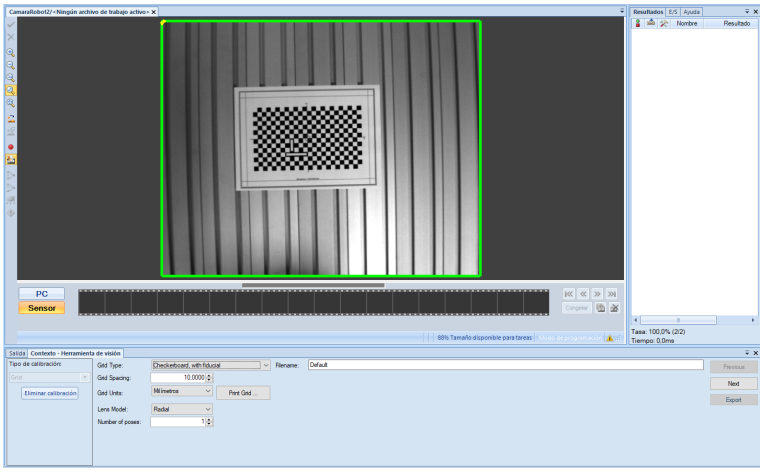


Figura 5.43: Tipo de calibración.

A continuación, se deberá presionar ”Next”, aparecerá una interfaz que calculará la posición de todas las celdas del tablero, cuando la función haya concluido deberían aparecer las celdas con un borde de color verde [Figura 5.44].

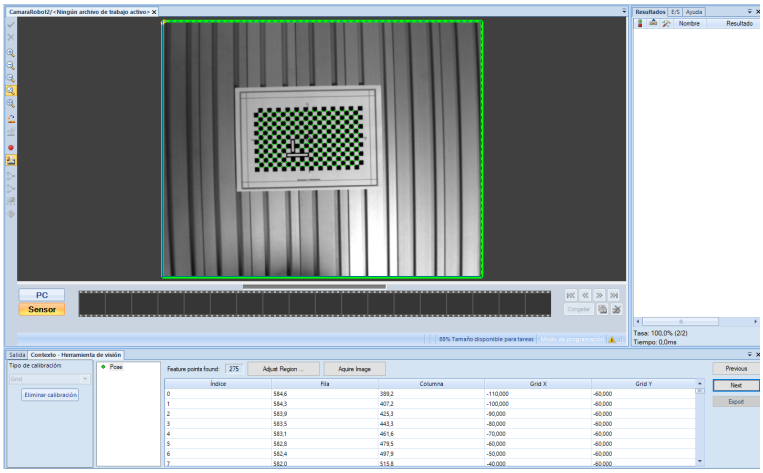
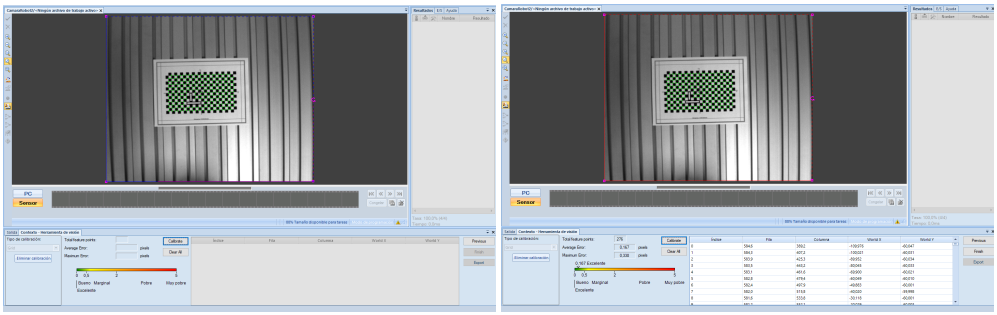


Figura 5.44: Cálculo de la posición de las celdas.

Finalmente, presionar ”Next” para dirigirse a la última fase del calibrado. Seleccionar el botón ”Calibrate” [Figura 5.45a] y esperar a la estimación de la calidad del calibrado. Una vez ha terminado la fase de calibrado, comprobar que la calidad entra dentro del rango marginal [Figura 5.45b], si es un buen calibrado finalizar la calibración y guardar el trabajo.



(a) Calibrar. **(b)** Calidad del calibrado.

Figura 5.45: Finalizar proceso de calibración.

5.5.3. Detección de objetos desde imágenes obtenidas en el entorno real

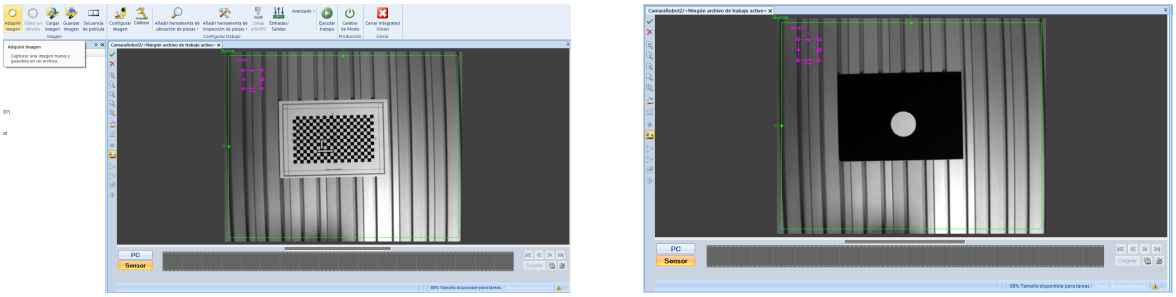
Una vez se ha conseguido una buena calibración, el siguiente paso es proceder a la detección de los objetos que va a manipular el robot. Para ello se va a utilizar una función denominada ”PatMax” de la que se ha obtenido información desde (Cognex, s.f.).

PatMax es una herramienta que comienza con imágenes de entrenamiento, en este proyecto serán imágenes captadas de los diferentes objetos que va a manipular el robot. Tras obtener las imágenes, la función las escanea en busca de bordes que identifican las formas geométricas. Estas muestras se convierten en clases que guardan información propia de las piezas, como posición y orientación del espacio de trabajo. Esta información digital se almacena en tablas de información, que la función memoriza y utiliza para diferenciar las piezas.

El primer paso para la detección de figuras en el programa es dirigirse a la ventana de ”Visión” y seleccionar ”Añadir herramienta de ubicación de piezas”. Aparecerán múltiples herramientas, en este proyecto como ya se ha comentado anteriormente se seleccionará ”Patrón PatMax” [Figura 5.46].

Figura 5.46: Herramienta PatMax.

El siguiente paso es adquirir una imagen del campo de visión de la cámara integrada [Figura 5.47a]. Para este proyecto cabe destacar que se ha colocado encima del grid de calibración una cartulina de color negra, para que las piezas que son de color blanco se puedan distinguir perfectamente del fondo [Figura 5.47b].



(a) Adquirir imagen. **(b)** Pieza dentro del espacio de trabajo.

Figura 5.47: Obtener una imagen de la pieza en el espacio de trabajo.

A continuación, como se puede observar, la herramienta tiene dos cuadrículas. La cuadrícula verde es la región de detección de la pieza y la cuadrícula morada es la región de detección del modelo que queremos entrenar. Se debe situar la cuadrícula morada encima de la pieza de manera que la cubra completamente. Una vez se hayan configurado las regiones de búsqueda y de modelo, presionar el botón aceptar [Figura 5.48].

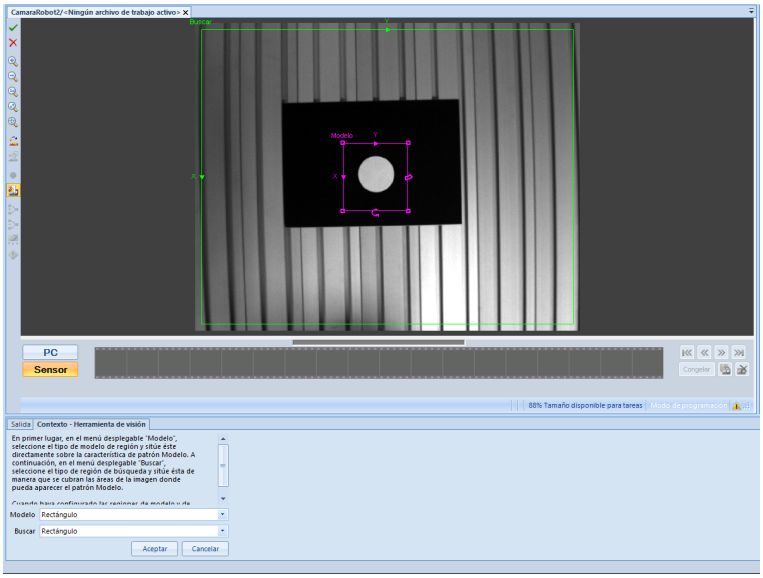


Figura 5.48: Regiones de búsqueda y modelo.

En el paso anterior se ha entrenado la herramienta con el modelo seleccionado, guardando las características propias de la figura. Posteriormente se debe elegir el nombre de la pieza con la que hemos entrenado la herramienta [Figura 5.49].

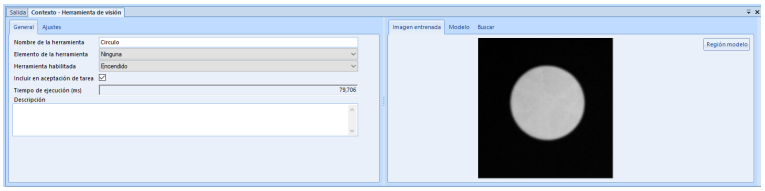


Figura 5.49: Ajustes generales de la pieza.

Seguidamente, se establece un umbral de detección para que la herramienta sea capaz de diferenciar las piezas con características similares, también se modifica la tolerancia a rotación para que pueda detectar figuras con diferentes orientaciones [Figura 5.50].

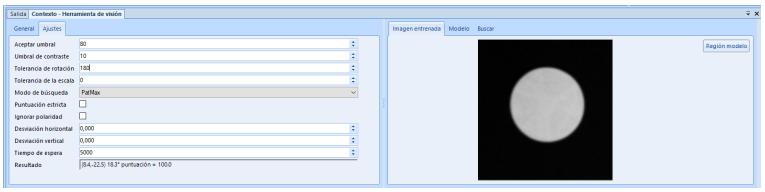


Figura 5.50: Ajustes específicos para la detección de la pieza.

Después de haber realizado estos ajustes, la herramienta es capaz de detectar las figuras con las que se ha entrenado dentro del campo de visión, sin importar la rotación o la posición de la pieza [Figura 5.51]. Es importante recalcar que en la ventana de resultados se muestra información referente a la pieza detectada, como tasa de acierto, velocidad de detección, rotación y posición en X e Y.

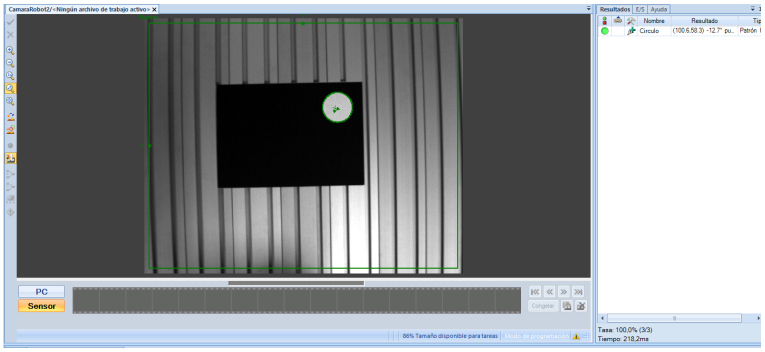


Figura 5.51: Pieza detectada dentro del campo de visión de la cámara.

Finalmente, para poder utilizar la información del modelo entrenado en el programa real y que el robot sea capaz de moverse a las posiciones de las piezas detectadas por la cámara, es necesario enviar los datos de las figura como salida a RAPID. Para ello, dirigirse a la ventana ”Visión” y seleccionar la opción ”Salida a RAPID”. Automáticamente aparecerá una ventana en la que se tendrán que especificar los datos que se enviarán al programa como variables en RAPID. Para este proyecto solamente es necesario tener conocimiento de la rotación en Z de la figura y de su posición en X e Y en el espacio de trabajo [Figura 5.52]. Para concluir, guardar el trabajo en el almacenamiento de la cámara.

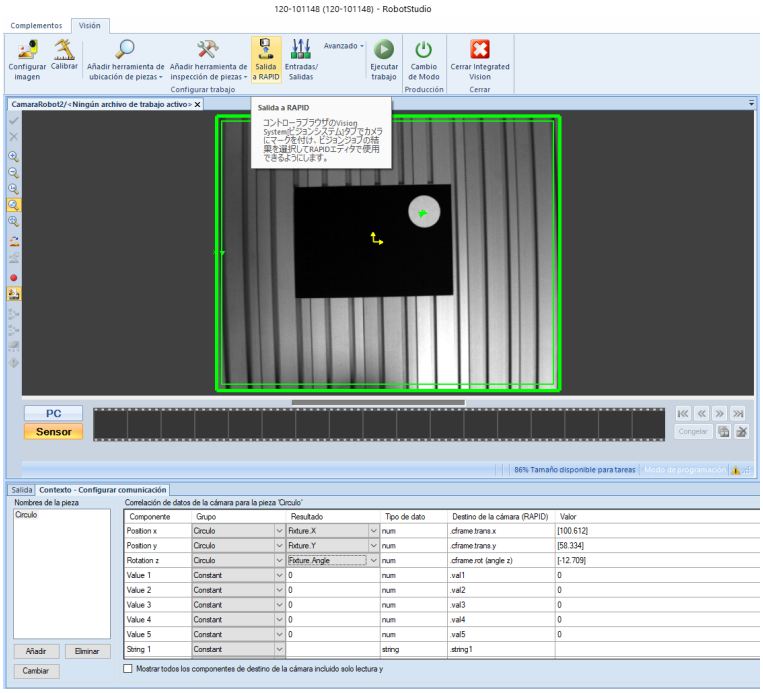


Figura 5.52: Enviar los datos de la pieza a RAPID.

5.6. Objetos de trabajo

Los objetos de trabajo del proyecto son las piezas con las que el robot va a interactuar, tanto en el entorno de simulación, como en el entorno real. Las principales diferencias son que en el entorno de simulación las piezas estarán situadas en un punto fijo de la mesa de trabajo. En cambio, en el entorno real, las piezas pueden variar su posición y será gracias al sistema de visión integrado que se podrá estimar la posición en la mesa de trabajo.

5.6.1. Objetos de trabajo en entorno de simulación

Los objetos de trabajo del entorno de simulación serán los archivos diseñados en FreeCad, los cuales se importarán y colocarán en el entorno de simulación de manera que posteriormente se pueda realizar una tarea de manipulación [Figura 5.53].

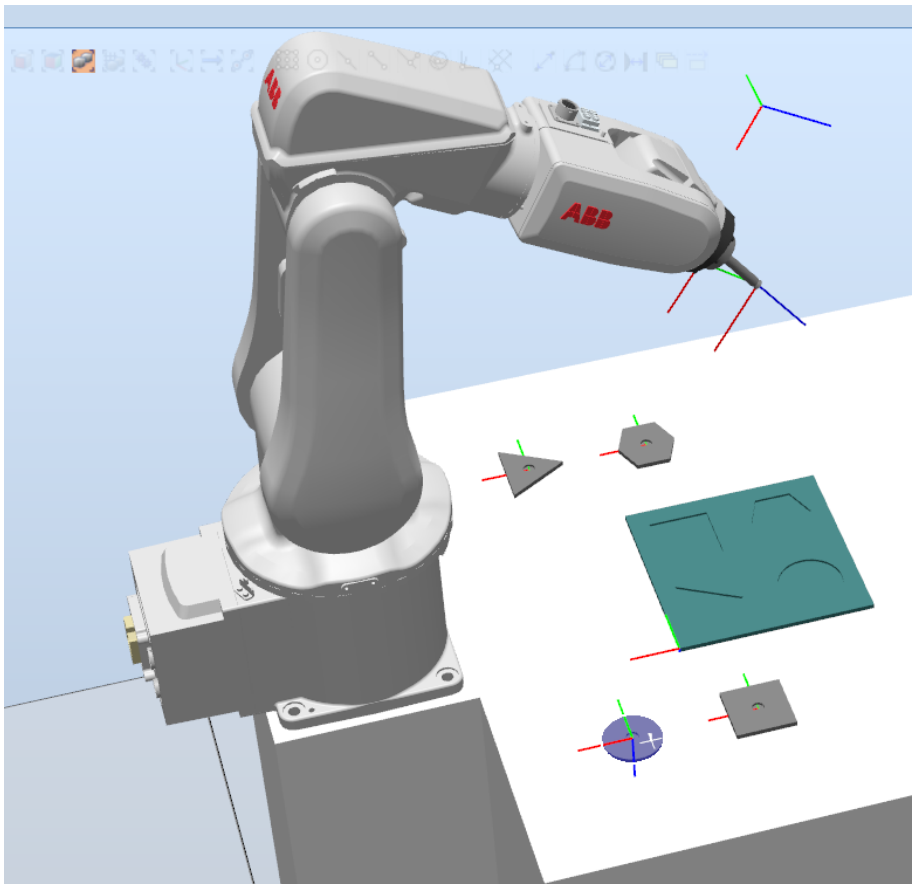


Figura 5.53: Mesa de trabajo con las piezas colocadas.

5.6.1.1. Importar geometrías 3D

Para importar un objeto diseñado en 3D previamente, desde la ventana "Modelado" utilizando la opción "Importar geometría" y seleccionando "Buscar geometría" [Figura 5.54], podremos acceder a la carpeta en la que se encuentran nuestras piezas diseñadas en 3D.

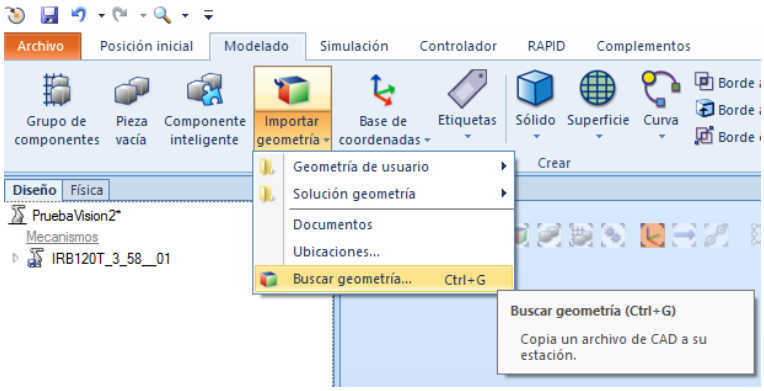


Figura 5.54: Importar geometría.

Desde la ventana que emerge dirigirse a la ruta donde se encuentran las piezas y seleccionarlas [Figura 5.55].

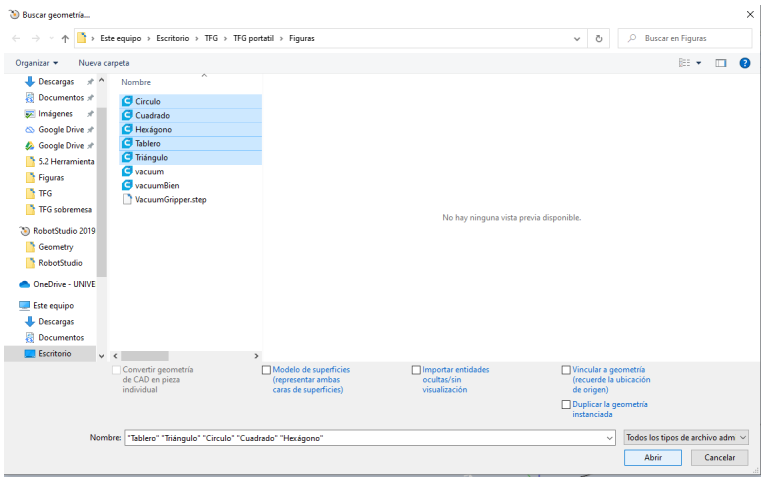


Figura 5.55: Seleccionar piezas a importar.

Lamentablemente, al cargar las piezas, estas aparecen en el entorno de simulación con unas posiciones que no son las adecuadas [Figura 5.56]. Debido a esto, se tiene que modificar su posición para colocarlas correctamente en la mesa de trabajo y establecer el origen local de la pieza.

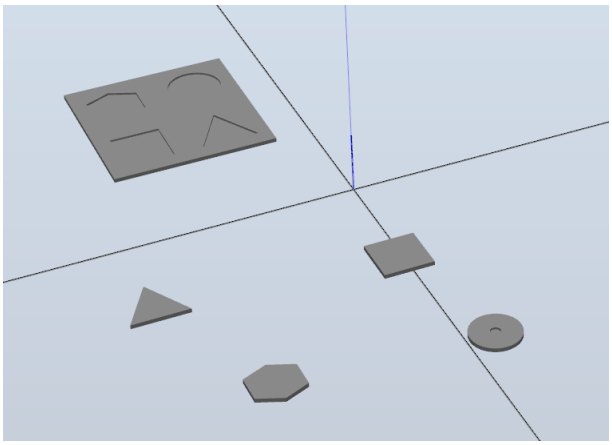


Figura 5.56: Piezas con posiciones erróneas.

5.6.1.2. Establecer el origen local y situar las piezas

Para poder colocar el objeto en la mesa de trabajo, primero debemos conocer las coordenadas [X,Y,Z] de la posición objetivo. Una vez se conoce la posición objetivo, seleccionar con el botón derecho del ratón la pieza que se requiere posicionar, utilizando la opción "Posición", y seleccionando "Fijar posición" [Figura 5.57], se podrá acceder a la ventana que modifica tanto la posición como la orientación de la pieza [Figura 5.58].

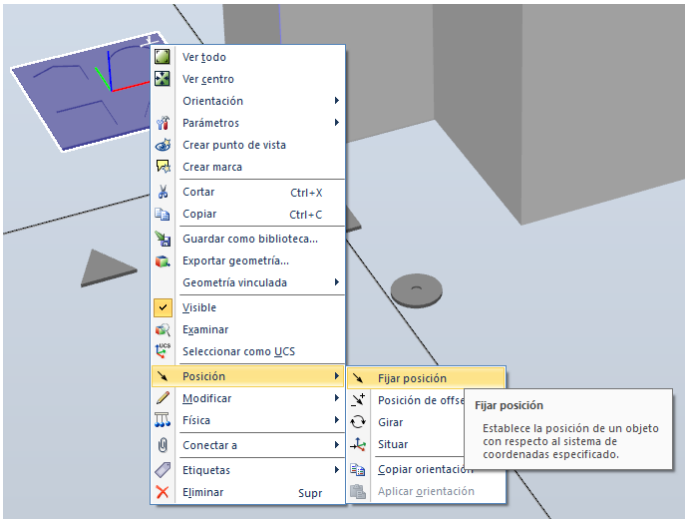


Figura 5.57: Seleccionar la herramienta "Fijar posición".

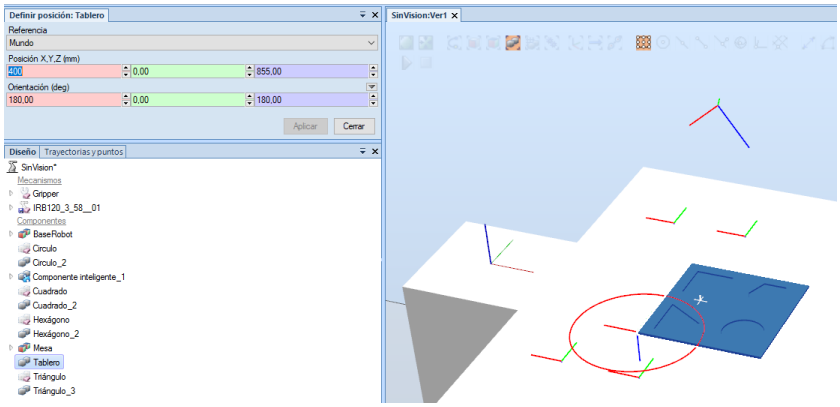
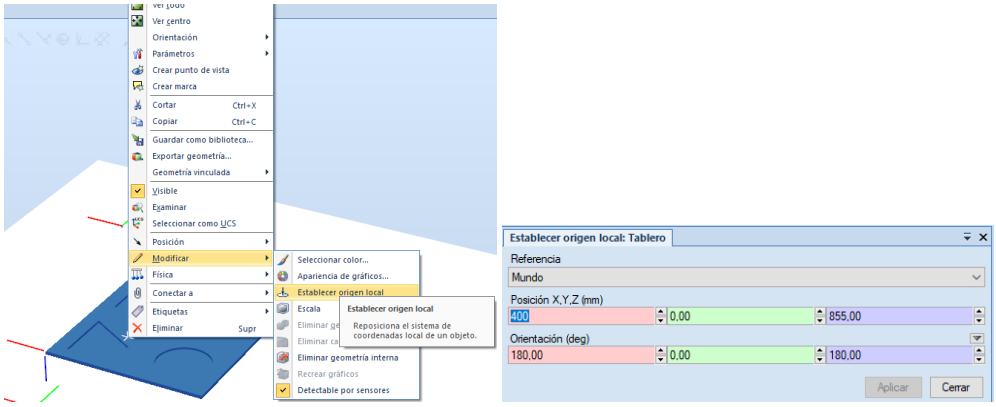


Figura 5.58: Definir posición de la pieza (Tablero).

Para establecer el origen local de un objeto, primero hay que conocer las coordenadas [X,Y,Z] y la orientación respecto al sistema de referencia "Mundo". Una vez se conoce la posición y la orientación del sistema de referencia local del objeto respecto del mundo, seleccionar con el botón derecho del ratón la pieza que se quiere modificar. Utilizando la opción "Modificar", y seleccionando "Establecer origen local" [Figura 5.59a], se podrá acceder a la ventana que modifica tanto la posición como la orientación del sistema de referencia local de la pieza [Figura 5.59b].



(a) Seleccionar la herramienta "Establecer origen local".

(b) Definir origen local de la pieza (Tablero).

Figura 5.59: Establecer origen local de una pieza.

Para nuestro entorno de simulación las piezas quedarían colocadas en la mesa de trabajo para que el robot las pueda manipular. Además, el sistema de referencia local de todas las piezas tiene la misma orientación, esto se ha realizado de esta manera para que en la tarea de manipulación coincida el eje Z de la herramienta con el de las piezas y la trayectoria de manipulación sea la óptima [Figura 5.60].

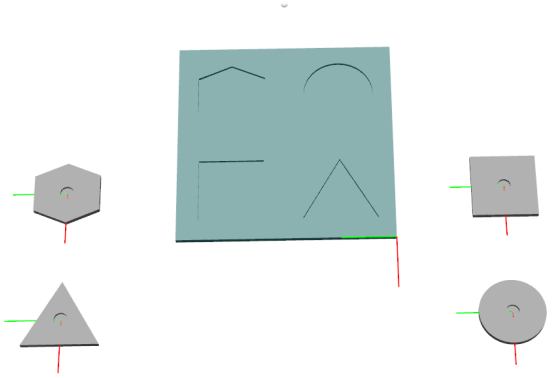


Figura 5.60: Posición y sistema de referencia local de las piezas en la mesa de trabajo.

5.6.1.3. Crear objetos de trabajo

Una vez se han posicionado las piezas y se ha establecido su origen local, se procede a crear el objeto de trabajo. El objeto de trabajo es un sistema de coordenadas que tiene que ser definido en dos bases de coordenadas: la base de coordenadas del objeto, que depende de la base de coordenadas del usuario, y la base de coordenadas del usuario, que depende de la base de coordenadas del mundo. Los objetos de trabajo se utilizan para simplificar la programación, ya que están asociados a las piezas que se van a manipular.

El primer paso para crear un objeto de trabajo es dirigirse a la ventana "Posición inicial", seleccionar la herramienta "Otros", y escoger la opción "Crear objeto de trabajo" [Figura 5.61].

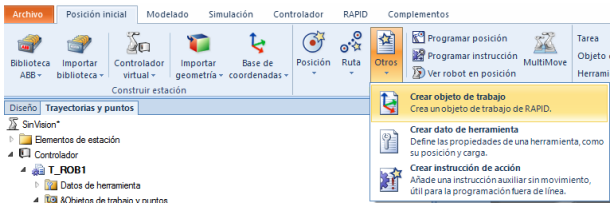


Figura 5.61: Herramienta crear objeto de trabajo.

Aparecerá una ventana en la que solamente se deberá modificar el nombre del objeto de trabajo y seleccionar el botón "Crear" [Figura 5.62].

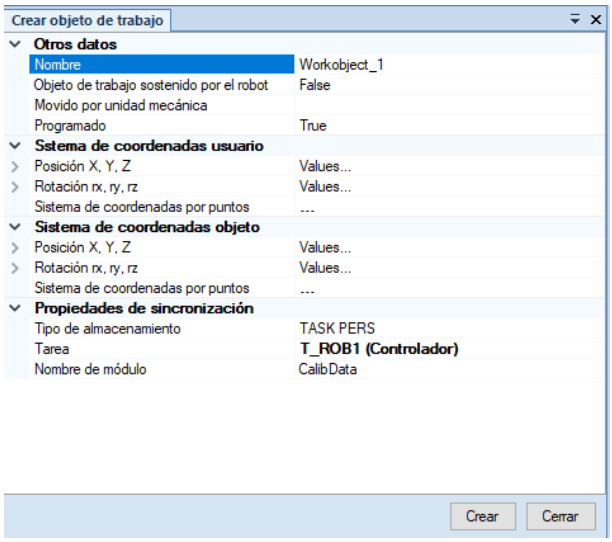


Figura 5.62: Crear objeto de trabajo.

Finalmente, en este proyecto se han creado seis objetos de trabajo, uno para cada pieza, otro para el tablero y el último el objeto de trabajo que hace referencia a la base del robot. Esto es así porque las posiciones de las piezas tienen que ser conocidas ya que no se dispone de la visión integrada en la simulación [Figura 5.63].

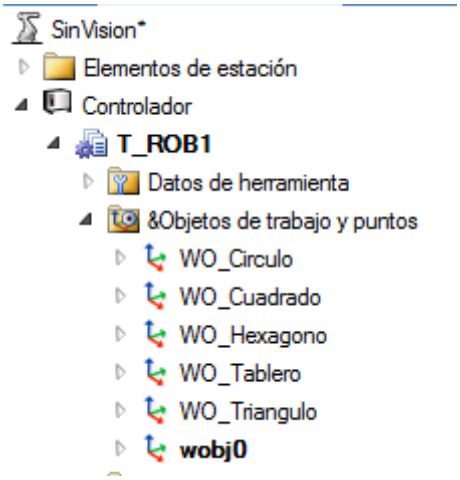
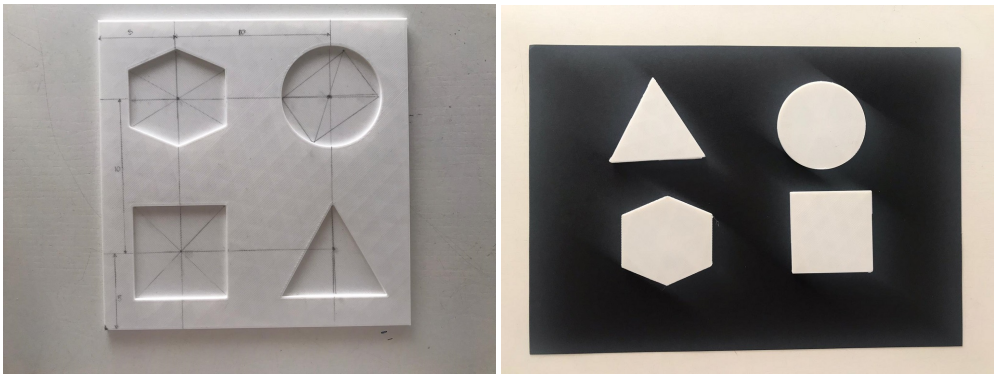


Figura 5.63: Lista de objetos de trabajo del proyecto.

5.6.2. Objetos de trabajo en entorno real

Los objetos de trabajo del entorno real tienen que tener posiciones conocidas en la mesa de trabajo, por lo que se definen en el FlexPendant, y su programación se explicará más adelante en la [Sección 5.7.2.5].

El entorno real estará compuesto por dos objetos de trabajo. El primero es el tablero, este objeto de trabajo se utiliza para simplificar la programación de la posición de dejada de las figuras. El segundo objeto de trabajo es la cartulina negra, el principal objetivo de este objeto de trabajo es establecer la zona de detección de las figuras y simplificar la posición de recogida de estas [Figura 5.64].



(a) Tablero. **(b)** Cartulina.

Figura 5.64: Objetos de trabajo del entorno real.

En cuanto a la disposición en la mesa de trabajo, la cartulina y el tablero están separadas del robot 20 centímetros, y entre ellas hay una distancia de 5 centímetros [5.65].

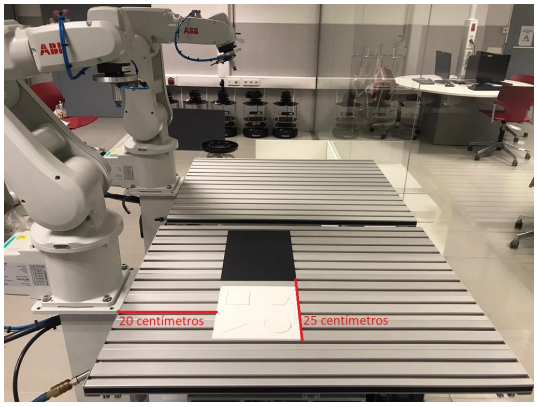


Figura 5.65: Distancia entre la zona de visión, el tablero y el robot.

Como en el entorno real la visión integrada estima la posición de las figuras dentro del objeto de trabajo cartulina, no es necesario crear un objeto de trabajo para cada pieza.

5.7. Programas desarrollados

Para la realización del proyecto se han desarrollado dos programas. El primero se ha ejecutado totalmente en el entorno de simulación de RobotStudio, hay que tener en cuenta que RobotStudio en el entorno de simulación no permite el uso de la programación en RAPID de la visión integrada, por lo que en este programa se prescinde de la cámara. El objetivo principal del programa en simulación es el pick and place de las piezas diseñadas en 3D, teniendo como conocimiento previo la posición de cogida y la posición de dejada de las figuras en la zona de trabajo.

El segundo programa se ha desarrollado en el entorno real, se ha utilizado el FlexPendant para definir tanto los objetos de trabajo como las posiciones objetivo del robot. Además, se ha empleado la cámara para obtener las imágenes de la zona de trabajo y detectar las piezas. Y por último, se ha realizado la programación en RAPID para que el robot ejecute las trayectorias pertinentes a las piezas detectadas. El objetivo principal del programa en el entorno real es el pick and place de las piezas detectadas por la cámara, esto se puede llevar a cabo gracias a los datos de posición y orientación que proporciona el sistema de visión durante la ejecución del programa.

5.7.1. Programación del entorno simulación

Una vez se han realizado las [Secciones 5.3, 5.6.1 y 5.2.1], la zona de trabajo en simulación estará lista para proceder a la programación de las posiciones y trayectorias del robot. Posteriormente, se configurarán las entradas y salidas digitales necesarias para poder utilizar la ventosa en esta parte del proyecto. Seguidamente, se procederá al desarrollo de la programación lógica de la estación empleando los componentes inteligentes que proporciona RobotStudio. Y finalmente, se realizará la programación pertinente del comportamiento del robot en la estación simulada utilizando la programación en RAPID.

5.7.1.1. Configurar posiciones

Las posiciones son los puntos objetivo en el espacio de trabajo a los cuales tiene que llegar el extremo del robot con una orientación definida. Estas posiciones dependen de los objetos de trabajo, es decir, una posición se creará teniendo en cuenta el objeto de trabajo con el que va a interactuar el robot.

Las posiciones definidas para desarrollar esta aplicación se pueden ver en la [Figura 5.66].

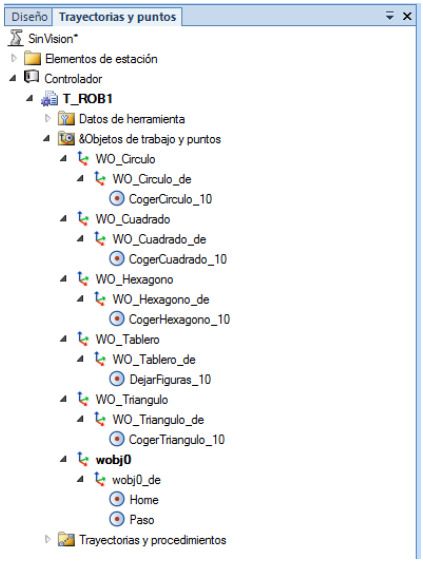


Figura 5.66: Posiciones del robot en el entorno de simulación.

Para crear una posición, dirigirse a la ventana "Posición inicial", seleccionar "Posición" y elegir la herramienta "Crear punto" [Figura 5.67].

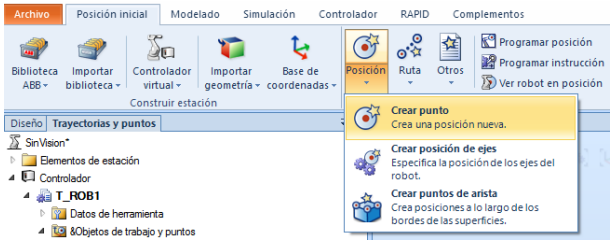


Figura 5.67: Herramienta crear punto.

Emergerá una ventana, en la cual hay que especificar la posición y orientación del punto respecto a un sistema de referencia. Cabe destacar que para este programa se ha utilizado como referencia "World" que es el sistema de referencia de la base del robot . También se debe modificar el nombre del punto y el objeto de trabajo correspondiente a cada figura al que va a estar ligado el punto [Figura 5.68]. Una vez modificados todos los datos, seleccionar el botón crear.

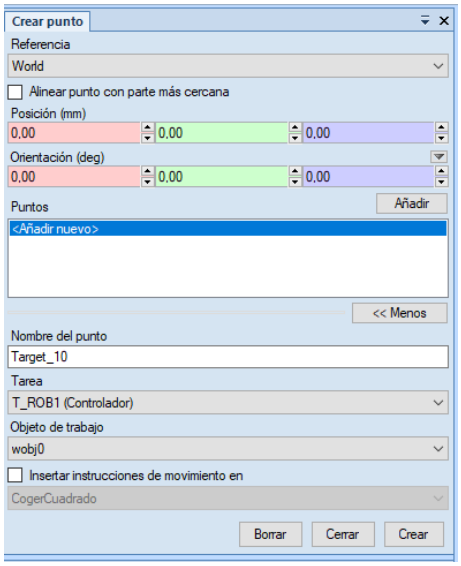


Figura 5.68: Crear punto.

Es importante que a la hora de crear un punto al cual tiene que llegar la herramienta de trabajo, se tenga en cuenta la orientación de los ejes del punto que se acaba de crear. Para realizar las tareas de manipulación y que la herramienta de trabajo pueda llegar a los puntos objetivo sin que el robot tenga que realizar movimientos o giros innecesarios, es necesario que tanto los ejes de la herramienta del robot como los del punto creado se alineen cuando el robot llegue a la posición definida.

Para ello, dirigirse al punto que se acaba de crear, seleccionarlo con el botón derecho del ratón y escoger la herramienta ”Modificar posición”, a continuación, escoger la opción ”Girar” [Figura 5.69].

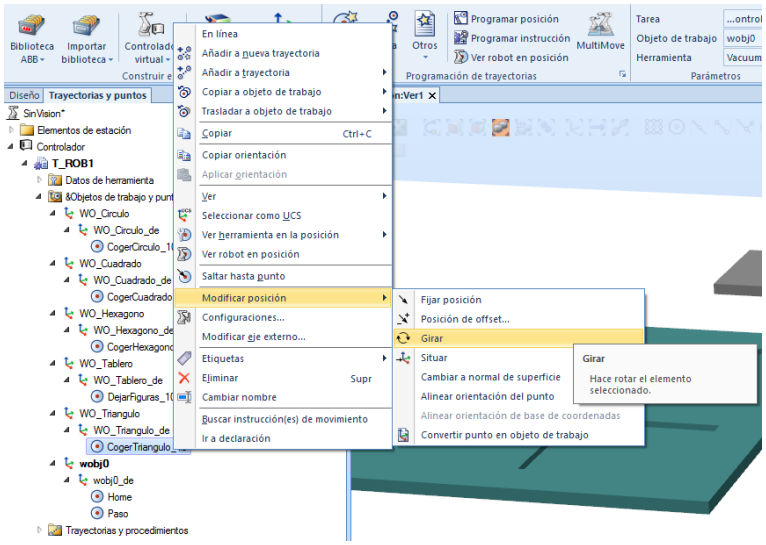


Figura 5.69: Herramienta girar.

Aparecerá una ventana en la cual se podrá modificar la orientación del sistema de referencia local del punto [Figura 5.70].

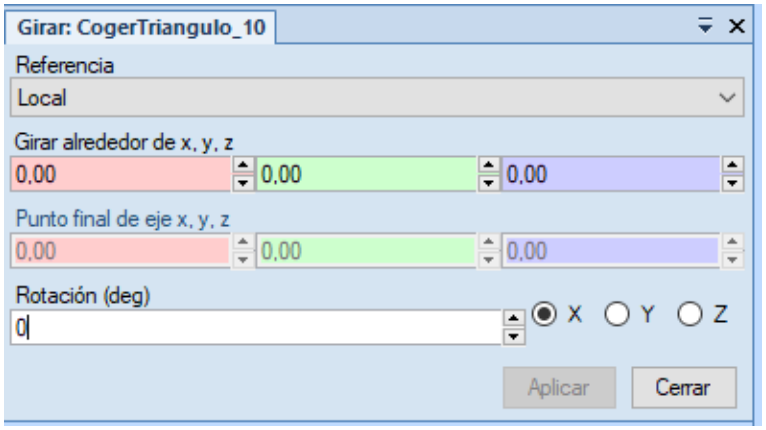


Figura 5.70: Girar punto.

Por último, para asegurarse de que el robot llega correctamente a los puntos definidos hay que comprobar las configuraciones articulares del robot en ese punto y escoger la más adecuada. La finalidad de esta modificación es eliminar movimientos innecesarios entre trayectorias.

Para cambiar las configuraciones articulares de un punto, primero seleccionamos con el botón derecho el punto deseado y escogemos la herramienta ”Configuraciones” [Figura 5.71].

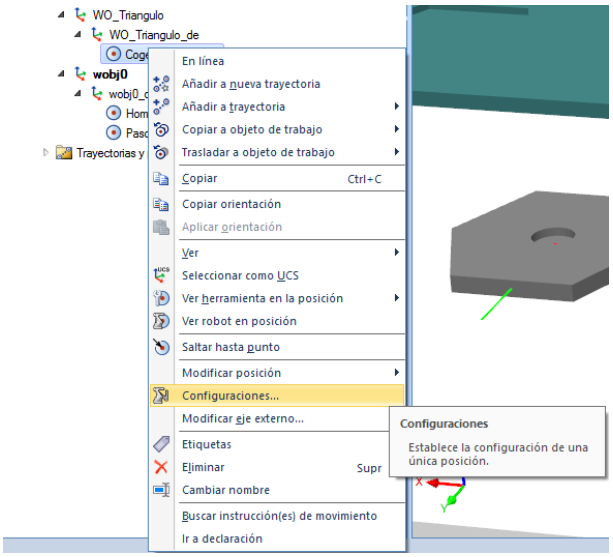


Figura 5.71: Configuraciones articulares en un punto.

En la ventana emergente seleccionar la configuración que tenga unos valores de posiciones articulares coherentes al punto que se esta intentando alcanzar [Figura 5.72].

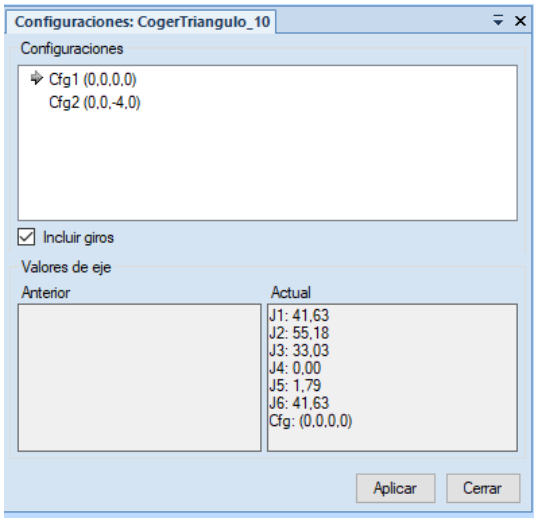


Figura 5.72: Seleccionar configuraciones articulares.

Una vez realizados todos los pasos, cuando el robot alcance el punto deseado se tienen que alinear los ejes de la herramienta de trabajo y de la posición objetivo. Además, el robot debería llegar con una configuración articular adecuada [Figura 5.73].

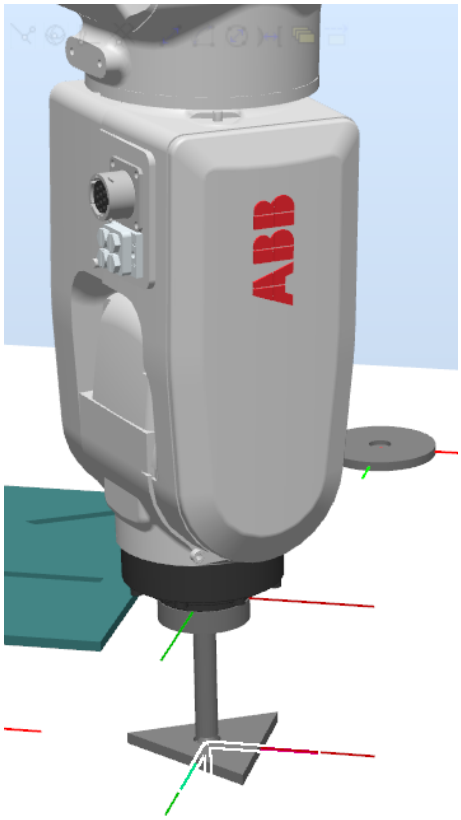


Figura 5.73: Alineación de los ejes de la herramienta y la posición objetivo.

5.7.1.2. Configurar trayectorias

Las trayectorias son el conjunto de movimientos que realiza el robot entre posiciones del espacio de trabajo. Los movimientos que realiza el robot pueden ser movimientos articulares o movimientos lineales y se puede modificar tanto la velocidad del movimiento como la zona de llegada al punto. Para realizar movimientos entre puntos que no requieren de precisión, como pueden ser puntos de paso, se utilizarán movimientos de tipo Joint y una zona de paso de 50 mm. Para realizar movimientos de cogida y dejada de piezas se necesitarán movimientos precisos y controlados por lo que se utilizarán movimientos de tipo Linear y una zona de paso fine.

Para crear una trayectoria se selecciona con el botón derecho la carpeta ”Trayectorias y procedimientos” y se escoge la opción ”Crear trayectoria” [Figura 5.74].

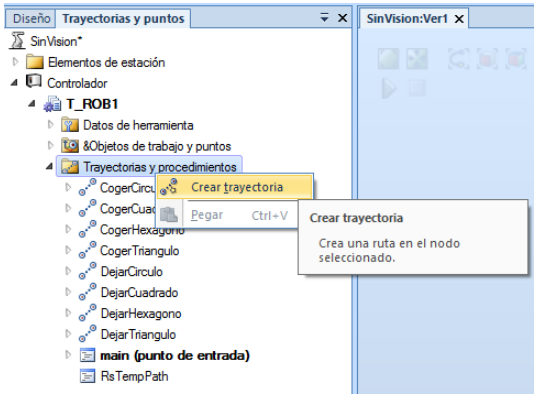
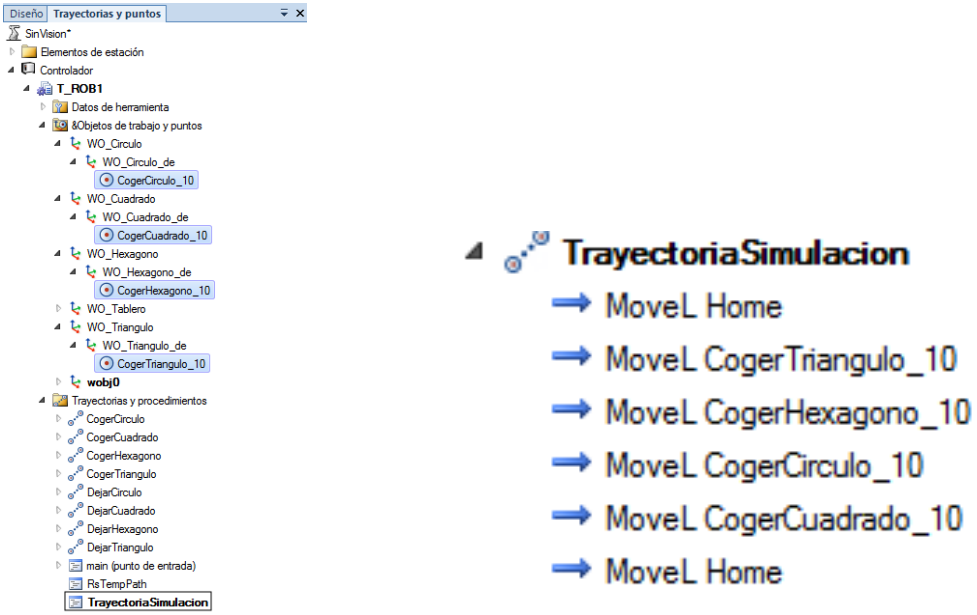


Figura 5.74: Herramienta crear trayectoria.

Una vez creada la trayectoria se le asigna un nombre, para crear un movimiento solo hay que arrastrar los puntos del espacio de trabajo a la trayectoria [Figura 5.75a]. Al añadir los puntos que conforman el movimiento que va a realizar el robot, la trayectoria final queda así [Figura 5.75b].



(a) Crear trayectoria. **(b)** Trayectoria creada

Figura 5.75

Para modificar los parámetros de los movimientos, se selecciona el movimiento dentro de una trayectoria con el botón derecho y se escoge la opción "Editar instrucción" [Figura 5.76].

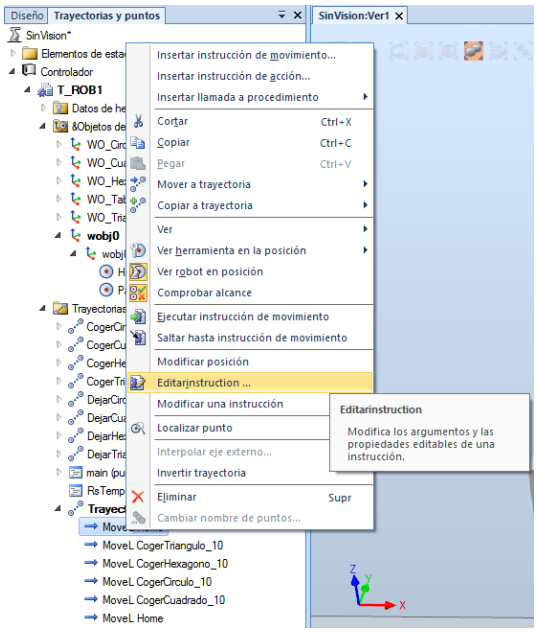


Figura 5.76: Herramienta modificar movimiento.

Aparecerá una ventana en la que se podrán modificar las propiedades del movimiento seleccionado, como el tipo de movimiento, la velocidad, la zona de paso y la herramienta con la que se va a llegar a los puntos objetivo de la trayectoria a ejecutar [Figura 5.77].

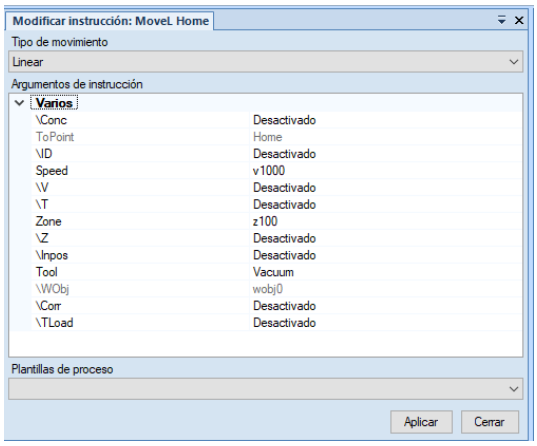


Figura 5.77: Modificar movimiento.

A continuación, se puede visualizar la trayectoria creada desde la ventana "Simulación" [Figura 5.78].

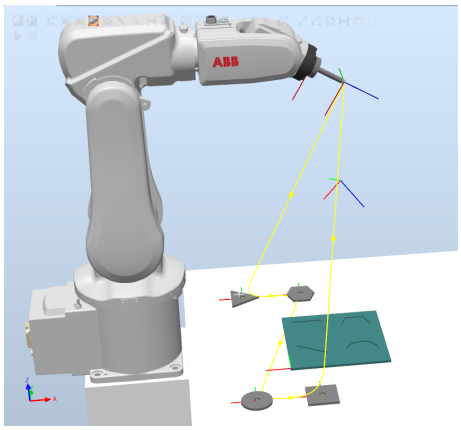


Figura 5.78: Visualizar trayectoria.

Finalmente, este proyecto tiene como objetivo realizar el pick and place de 4 figuras geométricas, por lo que se ha requerido crear 4 trayectorias para coger las piezas y 4 trayectorias para dejarlas en sus respectivas posiciones del tablero. En la siguiente [Figura 5.79] se muestran las trayectorias de pick and place del círculo con sus respectivos movimientos y el módulo Main que contiene todas las trayectorias que se ejecutan en secuencia.

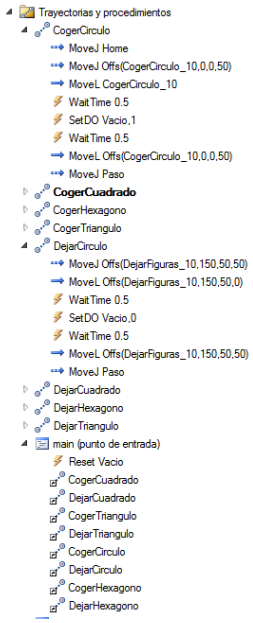


Figura 5.79: Trayectorias del proyecto en simulación.

5.7.1.3. Sincronizar con RAPID

Una vez se han configurado las trayectorias que va a realizar el robot hay que sincronizar la estación con RAPID. El objetivo es que todos los datos creados en la estación, como movimientos, puntos, objetos de trabajo y datos de la herramienta, puedan ser utilizados para desarrollar el comportamiento del robot en el entorno de programación de RAPID.

Para poder sincronizar la estación con RAPID dirigirse a la ventana "Posición inicial", seleccionar la herramienta "Sincronizar" y escoger la opción "Sincronizar con RAPID" [Figura 5.80].

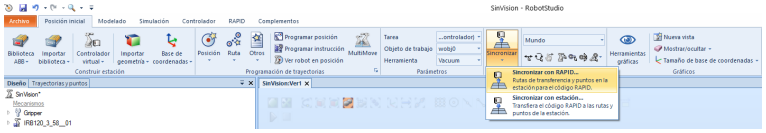


Figura 5.80: Herramienta sincronizar con RAPID.

Aparecerá una ventana en la que se tendrán que seleccionar los datos que deseamos transferir a RAPID [Figura 5.81]. Una vez seleccionados aplicar cambios seleccionando el botón "Aceptar".

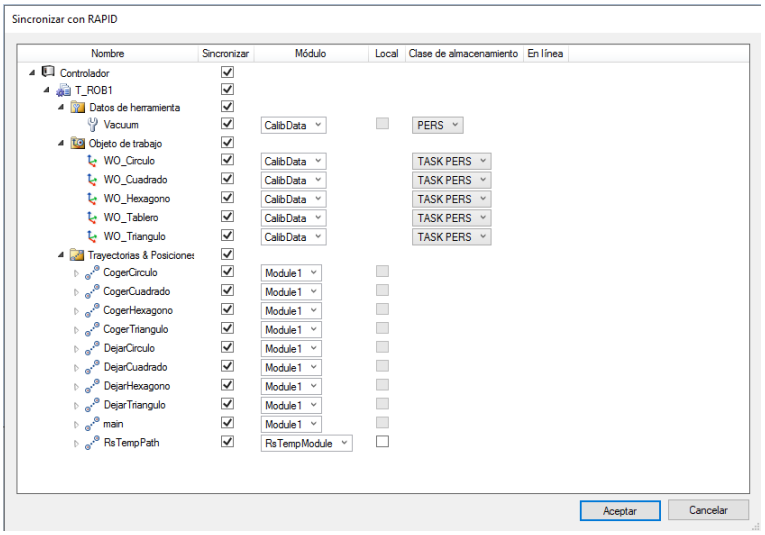


Figura 5.81: Transferir datos de la estación a RAPID.

5.7.1.4. Configurar entradas y salidas digitales

Para poder controlar la aspiración en el entorno de simulación es necesario crear una salida digital. Primeramente, dirigirse a la ventana "Controlador", seleccionar "Configuración" y escoger la opción "I/O System" [Figura 5.82].

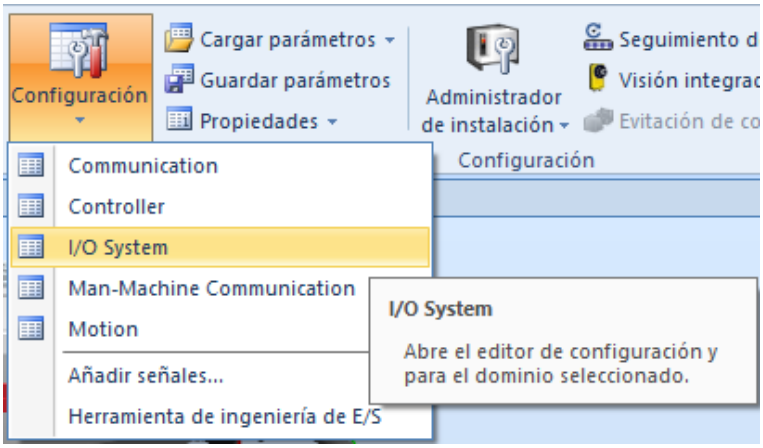


Figura 5.82: Herramienta I/O system.

A continuación, seleccionar con el botón derecho del ratón el "DeviceNet Device" y crear un nuevo controlador de E/S [Figura 5.83].

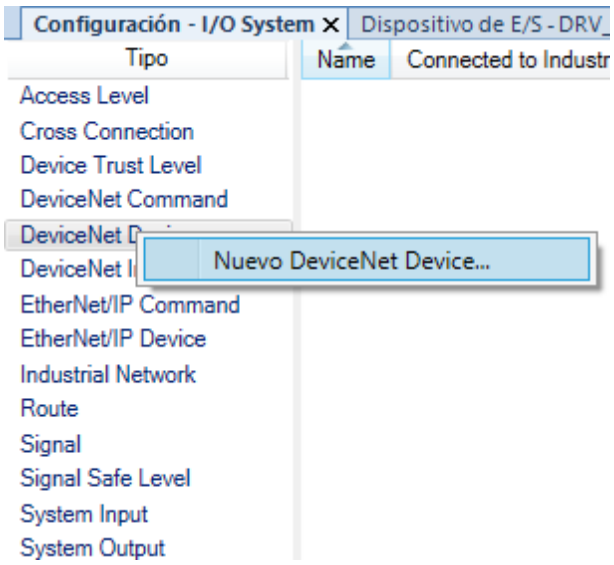


Figura 5.83: Crear controlador DeviceNet Device.

Emergerá una ventana, se deberá seleccionar el controlador de entradas y salidas lógicas, porque como se ha comentado anteriormente, sólo se va a crear una salida digital para controlar la activación y desactivación de la aspiración [Figura 5.84].

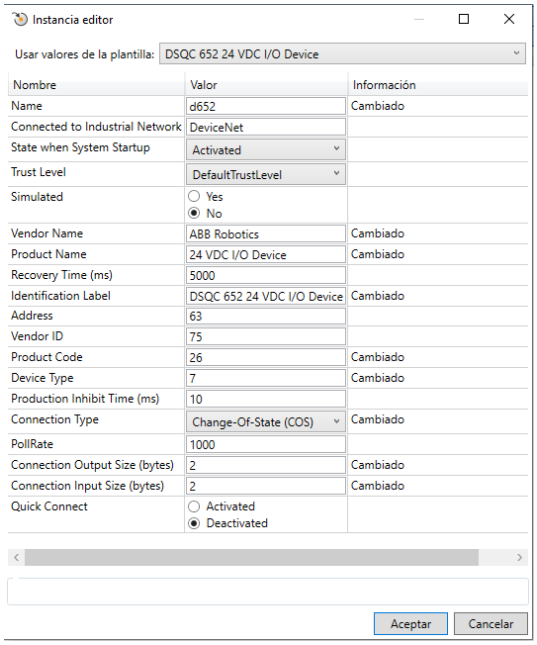


Figura 5.84: Seleccionar la tarjeta DSQC 652.

Una vez creado el controlador, se procede a crear la señal que controlará la aspiración. Seleccionar con el botón derecho del ratón la opción ”Signal” y crear una nueva señal [Figura 5.85].

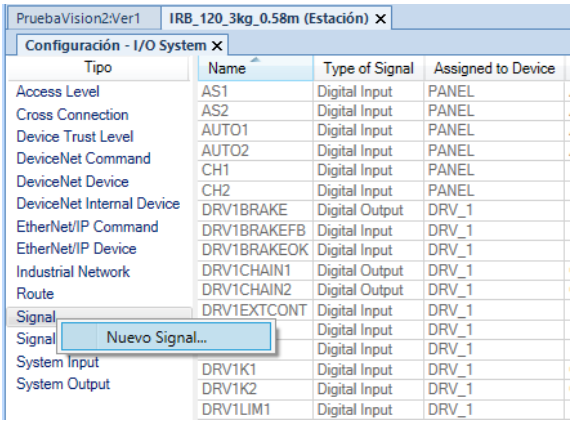


Figura 5.85: Herramienta crear señal.

Aparecerá una ventana en la que se podrá definir el nombre, el tipo, el dispositivo al que está ligado y el nivel de acceso de la señal [Figura 5.87].

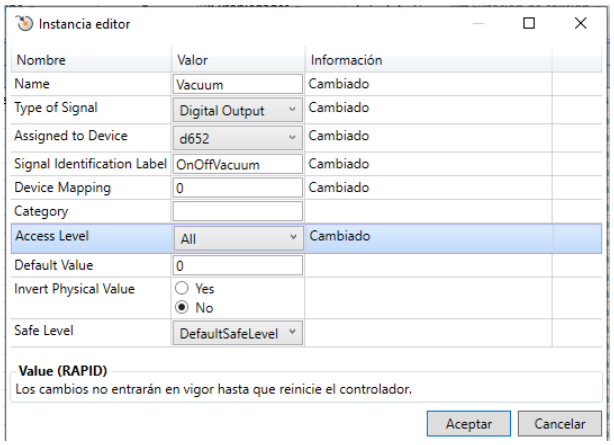


Figura 5.86: Crear señal de salida digital.

Finalmente, la señal de salida digital que controla la aspiración tiene que aparecer en el controlador de entradas y salidas lógicas [Figura 5.87].

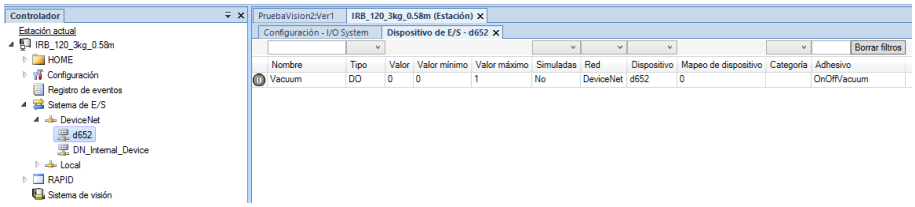


Figura 5.87: Visualización de la salida digital que controla la aspiración.

5.7.1.5. Componentes inteligentes

Los componentes inteligentes permiten establecer relaciones entre las señales del controlador y los elementos físicos que componen el entorno del robot. En este proyecto se van a utilizar para poder realizar la manipulación de las piezas, en concreto el momento en el que la ventosa succiona una figura y la transporta desde la zona de recogida a la zona de dejada.

Para poder crear un componente inteligente, primero hay que dirigirse a la ventana "Simulación", acceder a la opción "Lógica de estación" y entrar en la pestaña de "Diseño" [Figura 5.88].

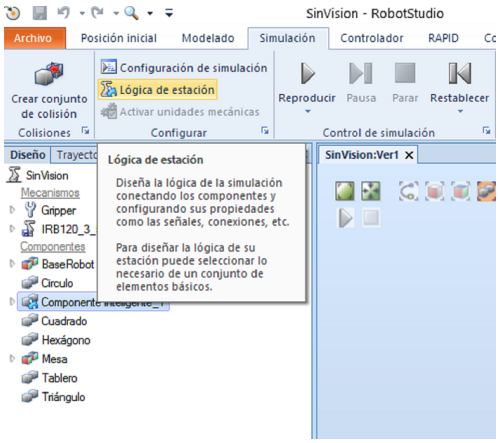


Figura 5.88: Acceder a la lógica de la estación.

Desde la ventana diseño, seleccionar el fondo con el botón derecho del ratón y escoger la opción "Componente inteligente vacío" [Figura 5.89].

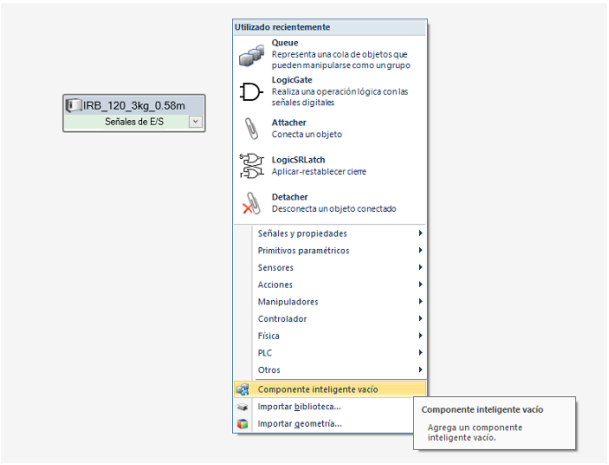


Figura 5.89: Crear un componente inteligente vacío.

Ahora deberá aparecer un bloque que hará referencia al controlador y otro que hará referencia al componente inteligente. A continuación, seleccionar la flecha desplegable de señales de entrada/salida del bloque del controlador y elegir la señal de salida digital creada en la [Sección 5.7.1.4] como se puede ver en la [Figura 5.90].

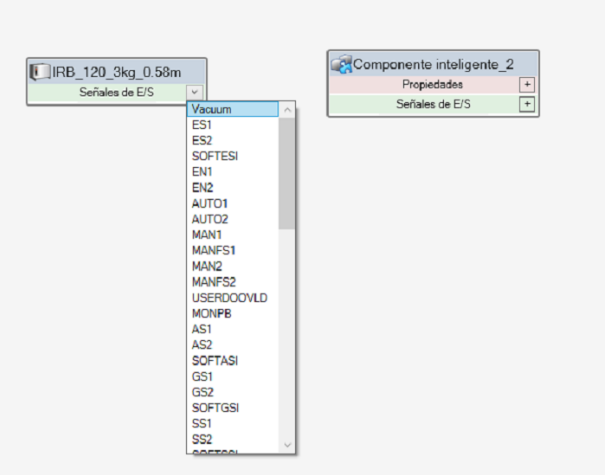


Figura 5.90: Seleccionar la salida digital "Vacuum" del controlador.

El siguiente paso es crear la señal de entrada del componente inteligente, para ello seleccionar la opción de señales de entrada/salida del bloque del componente inteligente. Emergerá una ventana en la que se podrá especificar el tipo y el nombre de la señal que se va a crear [Figura 5.91].

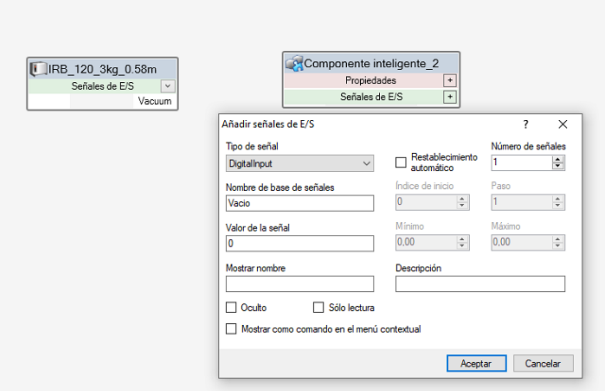


Figura 5.91: Crear la señal de entrada "Vacio" del componente inteligente.

Para establecer la relación entre la señal de salida del controlador ”Vacuum” y la señal de entrada ”Vacio” del componente inteligente, solo hay que unirlas desde los bloques de la ventana diseño [Figura 5.92].

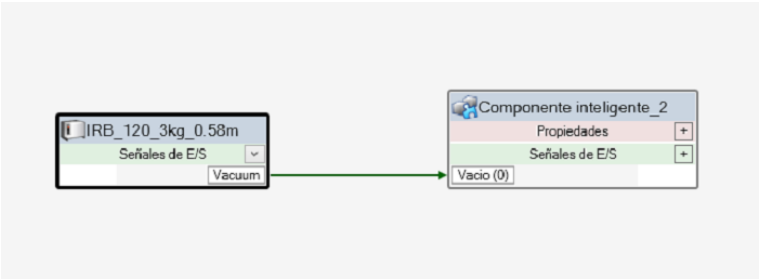


Figura 5.92: Enlazar las señales Vacuum y Vacio.

Al haber relacionado las señales que controlan el comportamiento de la estación, se procede a crear los bloques con los que se puede llevar a cabo este comportamiento. Para ello, hay que entrar en el bloque del componente inteligente [Figura 5.93].

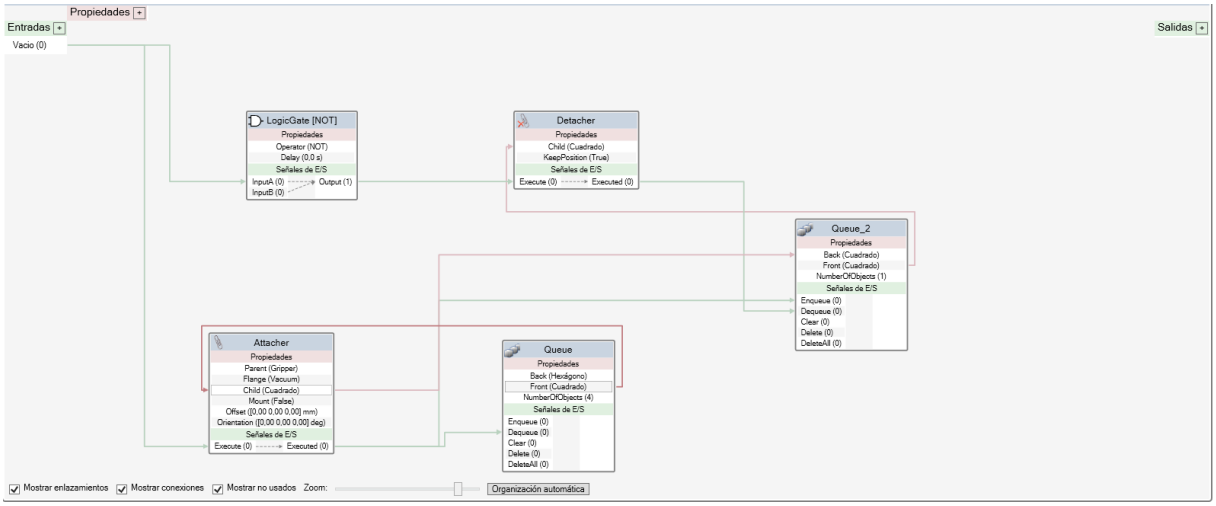


Figura 5.93: Bloques que forman el componente inteligente.

Como se puede observar, el comportamiento del programa depende de un diagrama de bloques compuesto de 5 bloques y de la señal de entrada del componente inteligente que es la que controla el comportamiento. Primero se va a explicar el funcionamiento general que tiene cada bloque de manera individual, y en la siguiente sección se explicará el funcionamiento específico que tiene el diagrama completo en el proyecto.

- Funcionamiento general de los bloques:
  1. Attacher: Conecta un objeto ”Child” a otro ”Parent” [Figura 5.94].

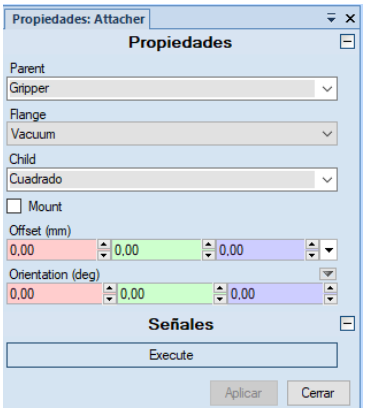


Figura 5.94: Attacher.

  2. Detacher: Desconecta un objeto [Figura 5.95].

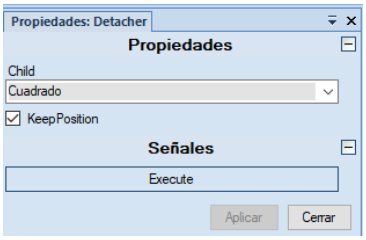


Figura 5.95: Detacher.

  3. Logic Gate [NOT]: Niega la entrada [Figura 5.96].

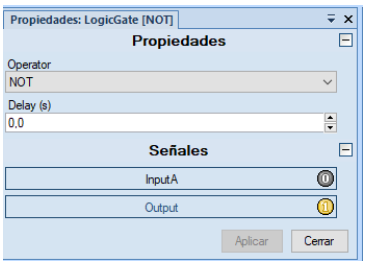


Figura 5.96: Puerta lógica NOT.

4. Queue: Representa una cola de objetos que pueden manipularse como un grupo, ”Front” es el objeto en la cabeza de la cola y ”Back” es el objeto al final de la cola [Figura 5.97].

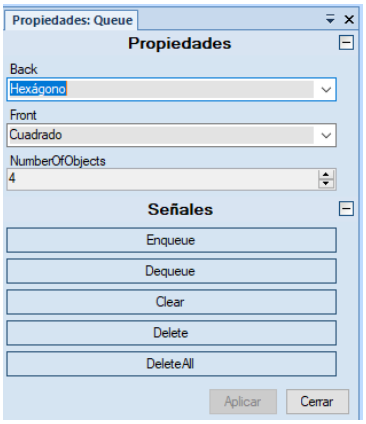


Figura 5.97: Cola.

5.7.1.6. Funcionamiento lógico del componente inteligente

El funcionamiento del componente inteligente depende de la señal de entrada ”Vacio”. Por eso se va a explicar el comportamiento del diagrama de bloques dependiendo del valor de esta señal.

Si la señal ”Vacio” vale 1, significa que el brazo robótico está en la posición de recogida de pieza o está en movimiento manipulando dicha pieza. Lo primero que hay que saber es que en el bloque ”Queue” se han encolado las figuras geométricas y el objeto que está en la posición ”Front” de la ”Queue” será el manipulado por el robot, es por eso que entra como ”Child” al ”Attacher”. Una vez se ejecuta el ”Attacher” al activarse la señal ”Vacio”, el objeto quedará conectado a la herramienta del robot ”Parent”. Además, el objeto que está conectado actualmente a la herramienta se encolará en ”Queue2” y será desencolado de ”Queue” para dar paso al siguiente objeto. De esta manera, el objeto que esta siendo manipulado actualmente por la herramienta quedará encolado en la posición ”Front” de la ”Queue2”, que está conectado directamente al ”Child” del ”Detacher”.

Cuando la señal ”Vacio” vale 0, el brazo robótico está en la posición de dejar pieza o no está manipulando ninguna figura. Esta señal cambia su valor con la ”puerta lógica NOT”, para poder activar el ”Detacher”, que realiza la desconexión del objeto con la herramienta de trabajo y desencola el objeto de la cola ”Queue2”.

Básicamente, cada vez que se activa la señal ”Vacio” la pieza a la cabeza de ”Queue” se conecta con la herramienta y pasa a la cola ”Queue2” a la espera de una señal de desconexión.

5.7.1.7. Programar comportamiento en RAPID

Al haber realizado la sincronización con RAPID en la [Sección 5.7.1.3], en el módulo de RAPID aparecerán todos los datos de posiciones y trayectorias para poder crear la función main que es la que ejecutará el comportamiento del programa.

Para acceder a los módulos de programación del proyecto, dirigirse a la ventana "RAPID", en la ventana "Controlador" aparecerá una pestaña llamada "RAPID" en la que aparecerán los módulos del programa [Figura 5.98].

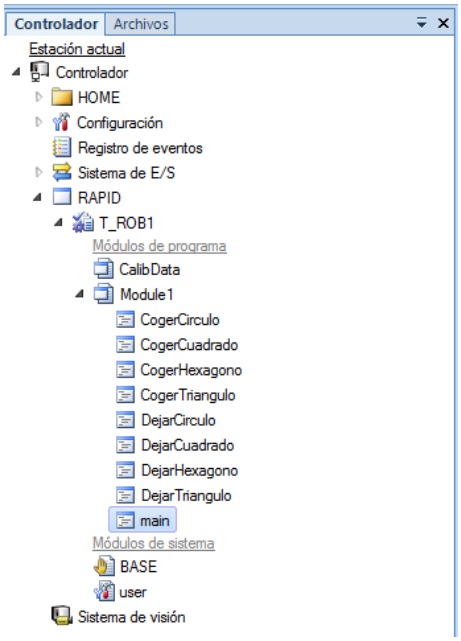


Figura 5.98: Módulos de programa.

En el módulo "CalibData", aparecerán los datos de los objetos de trabajo y de la herramienta de trabajo [Figura 5.99].

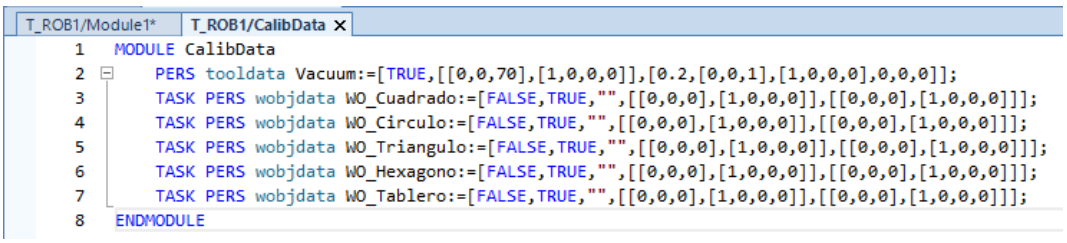
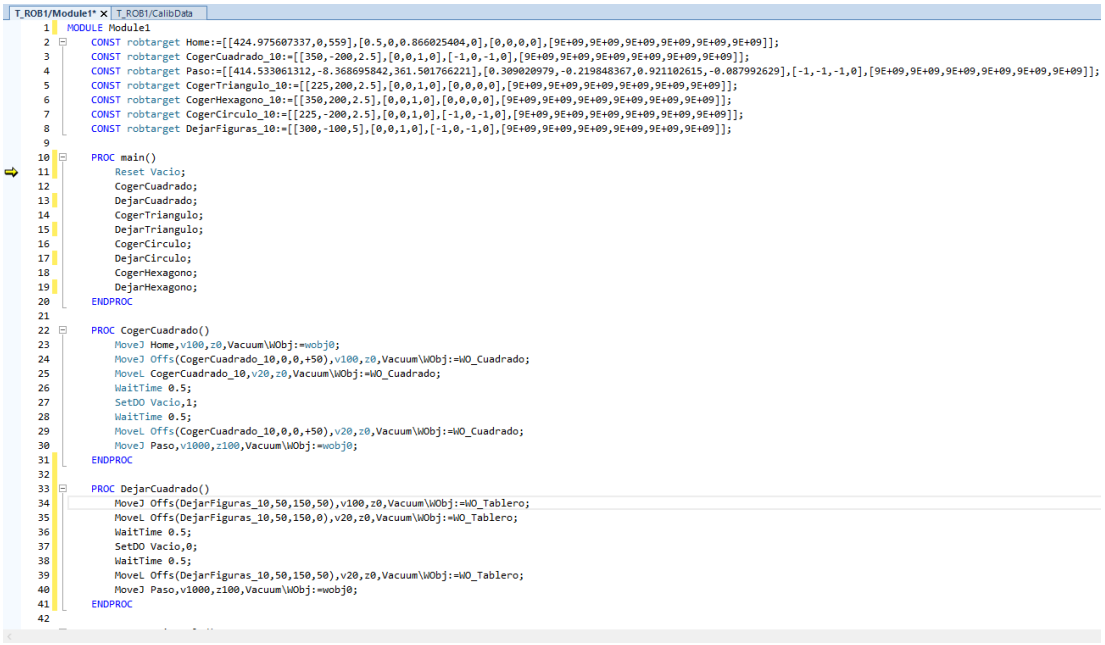


Figura 5.99: Módulo CalibData.

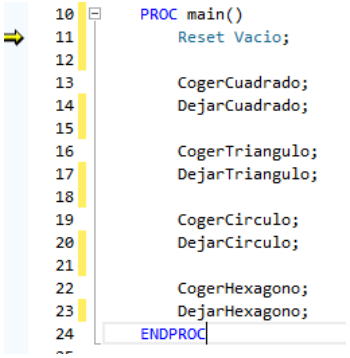
En el módulo ”Module1”, aparecerán todas las funciones que hacen referencia a las trayectorias creadas en la estación CogerPieza y DejarPieza. Además, también se encuentran la función main y las posiciones objetivo definidas en la [Sección 5.7.1.1], como se puede visualizar en la [Figura 5.100].



```
MODULE Module1
    CONST robtarget Home:=[[424.975607337,0,559],[0.5,0,0.866025404,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget CogerCuadrado_10:=[[350,-200,2.5],[0,0,1,0],[-1,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Paso:=[[414.533061312,-8.368695842,361.501766221],[0.309020979,-0.219848367,0.921102615,-0.087992629],[-1,-1,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget CogerTriangulo_10:=[[225,200,2.5],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget CogerHexagono_10:=[[350,200,2.5],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget CogerCirculo_10:=[[225,-200,2.5],[0,0,1,0],[-1,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget DejarFiguras_10:=[[300,-100,5],[0,0,1,0],[-1,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

    PROC main()
        Reset Vacio;
        CogerCuadrado;
        DejarCuadrado;
        CogerTriangulo;
        DejarTriangulo;
        CogerCirculo;
        DejarCirculo;
        CogerHexagono;
        DejarHexagono;
    ENDPROC

    PROC CogerCuadrado()
        MoveJ Home,v100,z0,Vacuum\WObj:=wobj0;
        MoveJ Offs(CogerCuadrado_10,0,0,+50),v100,z0,Vacuum\WObj:=WO_Cuadrado;
        MoveL CogerCuadrado_10,v20,z0,Vacuum\WObj:=WO_Cuadrado;
        WaitTime 0.5;
        SetDO Vacio,1;
        WaitTime 0.5;
        MoveL Offs(CogerCuadrado_10,0,0,+50),v20,z0,Vacuum\WObj:=WO_Cuadrado;
        MoveJ Paso,v1000,z100,Vacuum\WObj:=wobj0;
    ENDPROC

    PROC DejarCuadrado()
        MoveJ Offs(DejarFiguras_10,50,150,50),v100,z0,Vacuum\WObj:=WO_Tablero;
        MoveL Offs(DejarFiguras_10,50,150,0),v20,z0,Vacuum\WObj:=WO_Tablero;
        WaitTime 0.5;
        SetDO Vacio,0;
        WaitTime 0.5;
        MoveL Offs(DejarFiguras_10,50,150,50),v20,z0,Vacuum\WObj:=WO_Tablero;
        MoveJ Paso,v1000,z100,Vacuum\WObj:=wobj0;
    ENDPROC
```

Figura 5.100: Módulo Module1.

La función main, primero pone el valor de la variable ”Vacio” a 0, que es la que controla los componentes inteligentes. Después, llama en orden a las funciones CogerPieza y DejarPieza hasta que el robot haya realizado el pick and place de todas ellas [Figura 5.101].



```
PROC main()
    Reset Vacio;

    CogerCuadrado;
    DejarCuadrado;

    CogerTriangulo;
    DejarTriangulo;

    CogerCirculo;
    DejarCirculo;

    CogerHexagono;
    DejarHexagono;
ENDPROC
```

Figura 5.101: Función main.

La función CogerPieza, primero realiza un movimiento a ”Home”, después mueve el robot a la posición de aproximación de recoger pieza, que es un Offset de 5 cm en el eje z. A continuación, desplaza el robot hasta la posición de recogida, aplica un delay de 0.5 segundos y activa la señal ”Vacio” para conectar la pieza al extremo del robot. Finalmente, vuelve a realizar un movimiento al punto de aproximación y posteriormente al punto de ”Paso”.

Los movimientos a los puntos de ”Paso” y de ”Home” se realizan teniendo en cuenta como objeto de trabajo la base del robot ”wobj0”. En cambio, Los movimientos a los puntos de recogida y aproximación a la pieza se han realizado teniendo en cuenta como objeto de trabajo la propia pieza, en este caso ”WOCuadrado” porque la pieza es el cuadrado [5.102].

```
26  PROC CogerCuadrado()
27      MoveJ Home,v100,z0,Vacuum\WObj:=wobj0;
28      MoveJ Offs(CogerCuadrado_10,0,0,+50),v100,z0,Vacuum\WObj:=WO_Cuadrado;
29      MoveL CogerCuadrado_10,v20,z0,Vacuum\WObj:=WO_Cuadrado;
30      WaitTime 0.5;
31      SetDO Vacio,1;
32      WaitTime 0.5;
33      MoveL Offs(CogerCuadrado_10,0,0,+50),v20,z0,Vacuum\WObj:=WO_Cuadrado;
34      MoveJ Paso,v1000,z100,Vacuum\WObj:=wobj0;
35  ENDPROC
```

Figura 5.102: Función CogerPieza.

La función DejarPieza, primero realiza un movimiento al punto de aproximación de dejar la figura, que está a 5 cm en el eje z de la posición de dejada, después mueve el robot a la posición de dejada. Posteriormente, se aplica un delay de 0.5 segundos y se desactiva la señal ”Vacio” para desconectar la pieza del extremo del robot. Finalmente, vuelve a realizar un movimiento al punto de aproximación y posteriormente al punto de ”Paso”.

El movimiento al punto de ”Paso” se realiza teniendo en cuenta como objeto de trabajo la base del robot ”wobj0”. Sin embargo, los movimientos a los puntos de dejada y aproximación al punto de dejada, se han realizado teniendo en cuenta el objeto de trabajo ”Tablero” [5.103].

```
84  PROC DejarCuadrado()
85
86      MoveJ Offs(DejarFiguras_10,50,150,50),v100,z0,Vacuum\WObj:=WO_Tablero;
87      MoveL Offs(DejarFiguras_10,50,150,0),v20,z0,Vacuum\WObj:=WO_Tablero;
88      WaitTime 0.5;
89      SetDO Vacio,0;
90      WaitTime 0.5;
91      MoveL Offs(DejarFiguras_10,50,150,50),v20,z0,Vacuum\WObj:=WO_Tablero;
92      MoveJ Paso,v1000,z100,Vacuum\WObj:=wobj0;
```

Figura 5.103: Función DejarPieza.

Si se necesita visualizar el código completo del módulo ”Module1” dirigirse al [Código A.1].

5.7.2. Programación del entorno real

En esta sección se van a explicar todos los pasos que se han realizado para poder programar una tarea de manipulación en la célula robótica del entorno real.

5.7.2.1. Modo manual

Para poder programar el robot en el entorno real y controlarlo desde el FlexPendant, es necesario colocarlo en modo manual, ya que el modo de automático es el que se utiliza para poner en marcha y ejecutar los procesos automáticos.

Para ello, dirigirse al controlador del robot y girar en sentido horario la llave que se muestra en la siguiente [Figura 5.104]. A continuación, presionar el botón blanco de la derecha para rearmar los motores.

Figura 5.104: Colocar el robot en modo manual y encender los motores.

5.7.2.2. Señales de control de la herramienta de trabajo

El primer paso es establecer conexión con el controlador en línea [Sección 5.2.2.1] y después solicitar acceso de escritura desde RobotStudio [Figura 5.36].

Para poder realizar la programación es necesario saber cuáles son las señales que activan la herramienta de trabajo. Para ello desde RobotStudio conectado al controlador en línea, dirigirse a la ventana ”RAPID”, desplegar la pestaña de ”Señales de E/S” y seleccionar la tarjeta de control ”d652” [Figura 5.105].

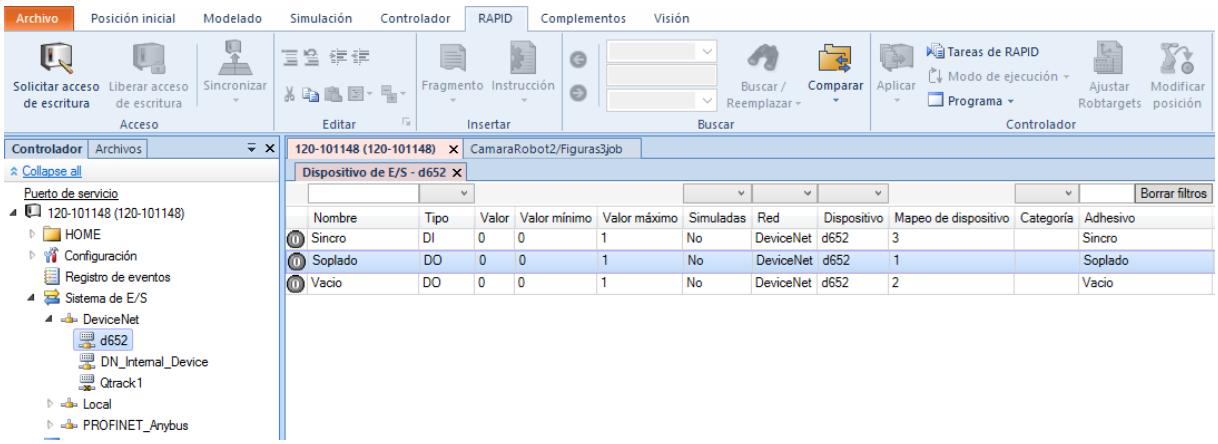


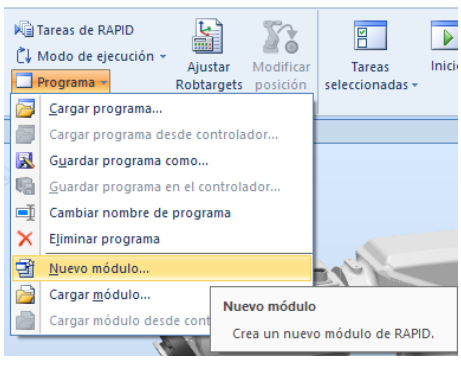
Figura 5.105: Visualizar las señales de control de la ventosa.

Como se puede comprobar, ya existen las salidas lógicas que controlan la herramienta de trabajo, las cuales son ”Soplado” que activa la aspiración de la ventosa y ”Vacio” que expulsa aire para poder dejar las piezas. Si fuese necesario se pueden crear más señales siguiendo las instrucciones de la [Sección 5.7.1.4], aunque no haría falta crear la tarjeta de control DeviceNet, ya que el controlador real la tiene instalada por defecto.

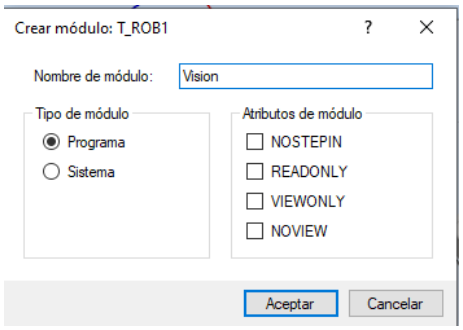
5.7.2.3. Programar el módulo principal

Una vez se ha establecido el modo de trabajo manual y se han comprobado las señales de control de la herramienta de trabajo, se puede proceder a la programación del comportamiento del robot en el entorno real. Posteriormente, se modificarán las variables que dependen del entorno del robot (robtargets y workobjects), utilizando el FlexPendant.

Primeramente, dirigirse a la ventana ”RAPID”, seleccionar ”Programa” y escoger la opción ”Nuevo módulo” [Figura 5.106].



(a) Herramienta crear nuevo módulo



(b) Establecer el nombre del módulo de programa.

Figura 5.106: Crear nuevo módulo de programa.

A continuación, en el módulo creado se procede a declarar las variables que se van a utilizar en el programa [Figura 5.107].

```
!Declarar el trabajo de visión que se va a utilizar en el programa
CONST string myjob := "Figuras3.job";
VAR cameratarget mycameratarget;

!Declarar herramienta de trabajo
PERS tooldata Vacuum:=[TRUE,[[0,0,60],[1,0,0,0]],[0.3,[0,0,1],[1,0,0,0],0,0,0]];

!Declarar los objetos de trabajo
TASK PERS wobjdata WobjGrid2:=[FALSE,TRUE,"",[[426.165,163.188,22.9781],[0.708151,-0.00707719,-0.00616661,-0.705998]],[[5.91058,9.13198,0],[0.999839,0,0,0.0179505]]];
TASK PERS wobjdata Tablero:=[FALSE,TRUE,"",[[326.303,-22.277,32.1978],[0.708382,-1.18486E-05,2.39246E-05,-0.705829]],[[0,0,0],[1,0,0,0]]];

!Declarar los robot target
CONST robtarget HOME:=[[136.59,-0.20,498.47],[0.00818261,0.904813,-0.425624,0.00955078],[0,-1,2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget myrobtarget:=[[-0.05,2.65,2.83],[0.0048476,-0.948875,-0.31559,-0.00407687],[0,-1,2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget DejarPieza:=[[-0.27,-0.56,-0.71],[0.00115169,-0.94137,-0.337146,-0.0123687],[-1,-1,2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget PASO:=[[405.44,17.41,114.63],[0.00812611,0.904814,-0.425621,0.00966605],[0,-1,2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

Figura 5.107: Declarar variables.

Lo siguiente es crear la función MoveToDetectedObject [Figura 5.108] y la función main [5.109]

```
PROC MoveToDetectedObject()

    !Seleccionar y activar la cámara que tenemos conectada con el trabajo de visión
    CamSetProgramMode CamaraRobot2;
    CamLoadJob CamaraRobot2, myjob;
    CamSetRunMode CamaraRobot2;

    !Adquirir una imagen de la zona de detección de figuras y guardarla
    CamReqImage CamaraRobot2;
    CamGetResult CamaraRobot2, mycameratarget;
    WobjGrid2.oframe := mycameratarget.cframe;

    !Mover el robot hacia la zona de aproximación de manipulación de la figura
    MoveL Offs(myrobtarget,0,0,50), v100, fine, Vacuum\WObj:=WobjGrid2;
    !Mover el robot hacia la zona de manipulación de la figura
    MoveL myrobtarget, v100, fine, Vacuum\WObj:=WobjGrid2;

    !Activar la aspiración de la ventosa
    WaitTime 0.5;
    SetDo Soplado,1;
    WaitTime 0.5;

    !Dependiendo de la figura detectada llamar a la función que las deja en su posición correspondiente
    IF mycameratarget.name = "Circulo" THEN
        DejarCirculo;
    ELSEIF mycameratarget.name = "Cuadrado" THEN
        DejarCuadrado;
    ELSEIF mycameratarget.name = "Poligono" THEN
        DejarHexagono;
    ELSEIF mycameratarget.name = "Triangulo" THEN
        DejarTriangulo;
    ENDIF

ENDPROC
```

Figura 5.108: Función MoveToDetectedObject.

```
55  PROC Main()
56      !Mover el robot a la posición inicial HOME
57      MoveJ HOME, v100, fine, Vacuum;
58
59      !Llamar a la función que adquiere una imagen de la zona de detección y manipula las piezas
60      MoveToDetectedObject;
61
62      !Mover el robot a la posición final HOME
63      MoveJ HOME, v100, fine, Vacuum;
64
65  ENDPROC
```

Figura 5.109: Función Main.

Cómo se puede comprobar entre las líneas [43,51], existen cuatro funciones distintas para dejar las piezas en sus respectivas posiciones. Todas ellas tienen una estructura similar aunque difieren en la posición de dejada, ya que en cada función se aplica un Offset diferente a la posición objetivo "DejarPieza" [Figura 5.110].

```
67  PROC DejarCuadrado()
68
69      !Mover el robot a la posición de paso entre la zona de detección y el tablero
70      MoveJ PASO, v100, fine, Vacuum;
71
72      !Mover el robot hacia la zona de aproximación de dejar la figura
73      MoveJ Offs(DejarPieza,50,50,25), v100, fine, Vacuum\WObj:=Tablero;
74
75      !Mover el robot hacia la zona de dejar la figura
76      MoveL Offs(DejarPieza,50,50,0), v20, fine, Vacuum\WObj:=Tablero;
77
78      !Desactivar la aspiración de la ventosa
79      WaitTime 0.5;
80      SetDo Soplado,0;
81      WaitTime 0.5;
82
83      !Mover el robot hacia la zona de aproximación de dejar la figura
84      MoveL Offs(DejarPieza,50,50,25), v100, fine, Vacuum\WObj:=Tablero;
```

Figura 5.110: Función DejarPieza.

A la hora de la ejecución del código es la función main la que tiene el puntero al programa. En cuanto al comportamiento, el robot comienza en la posición "HOME". A continuación, se activa la cámara, se adquiere una imagen y se procesa para detectar cualquier figura que se encuentre en la zona de detección. Posteriormente, el robot se mueve hacia la posición que ha estimado de la figura detectada y la manipula activando la ventosa. Dependiendo de la figura que se esté manipulando, el robot debe dejarla en una posición diferente del tablero, es por eso que se distingue la pieza mediante una sentencia de control. Finalmente, el robot utiliza la posición "PASO" para aproximarse a la zona de dejada, deja la pieza en su posición correspondiente desactivando la ventosa y vuelve a la posición inicial "HOME".

Si se necesita visualizar el código completo del módulo "Visión" dirigirse al [Código B.1].

5.7.2.4. Transferir el programa del PC al FlexPendant

Una vez realizada la programación desde el PC conectado en línea al controlador, se debe aplicar el programa al robot real. La finalidad de esta tarea es que aparezca el programa en el FlexPendant y poder modificar posteriormente las posiciones y objetos de trabajo que se necesitan para llevar a cabo el proceso automático.

El primer paso es dirigirse a la ventana "RAPID", acceder al módulo de programa "Vision" y seleccionar "Aplicar" [Figura 5.111].

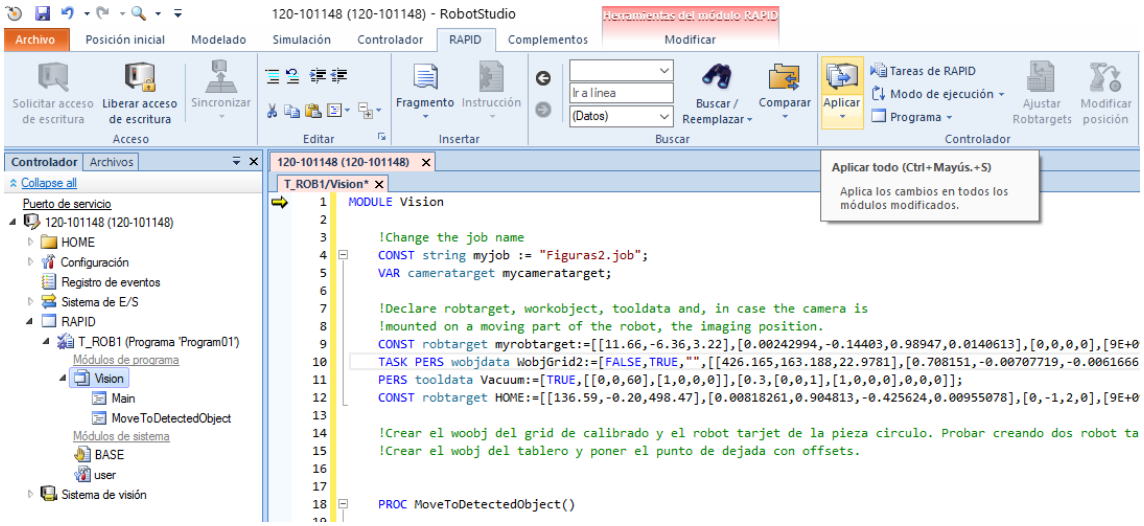
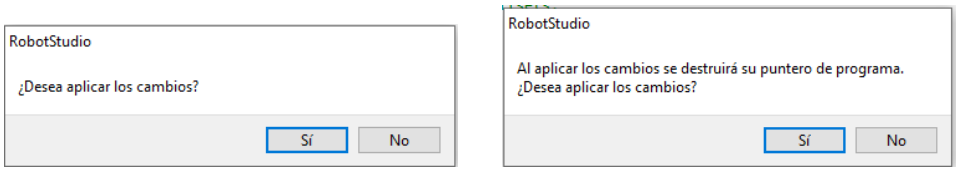


Figura 5.111: Transferir el programa desde el PC al robot real.

A continuación, emergerán dos ventanas, al aceptarlas se transferirán los módulos de programa al robot real [Figura 5.112].



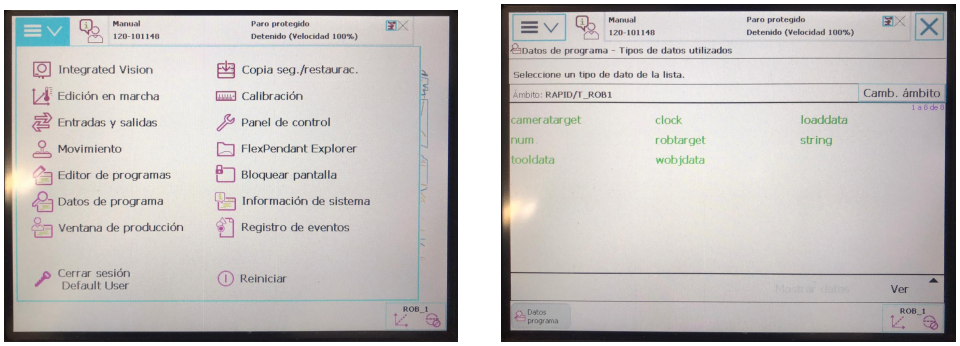
(a) Aplicar cambios. **(b)** Sobreescribir el programa.

Figura 5.112: Aceptar la transferencia del módulo de programa al robot real.

5.7.2.5. Definir los WorkObjects

En la [Sección 5.6.2] se han comentado las especificaciones y las posiciones que tendrán tanto el "Tablero" como la "Cartulina" en la zona de trabajo. En la [Sección 5.7.2.3] se ha descrito como se han inicializado los objetos de trabajo en el código del programa. Por último, en esta sección se explicará como definir el valor de los objetos de trabajo del entorno real en el programa.

Los objetos de trabajo en el entorno real se deben programar utilizando el FlexPendant, por lo que es necesario que el robot se encuentre en modo de trabajo manual. El primer paso es coger el FlexPendant y desde "Menú", seleccionar la opción "Datos de programa" [Figura 5.113].



(a) Menú FlexPendant. **(b)** Datos de programa FlexPendant.

Figura 5.113: Interfaz del FlexPendant.

En la ventana "Datos de Programa", aparecerán todos los tipos de datos de las variables que afectan a la programación del entorno real. En este caso, se seleccionará "wobjdata" [Figura 5.114].

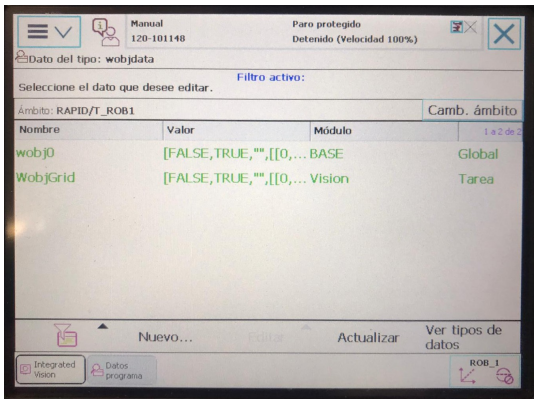


Figura 5.114: Ventana wobjdata del FlexPendant.

A continuación, se mostrarán todos los objetos de trabajo inicializados en la [Sección 5.7.2.3], seleccionar el objeto que requiera modificarse, elegir la opción "Editar" y presionar en el botón "Definir" [Figura 5.115].

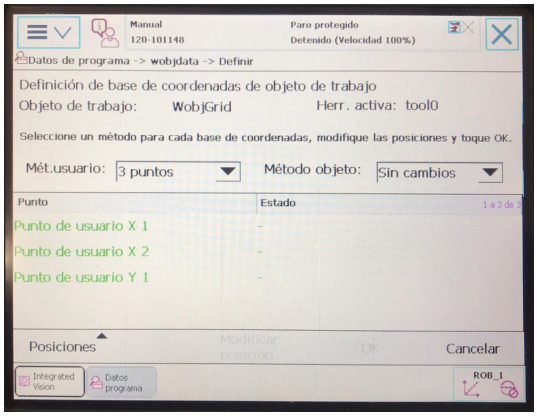
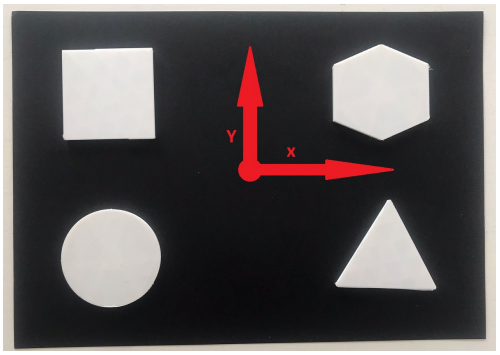


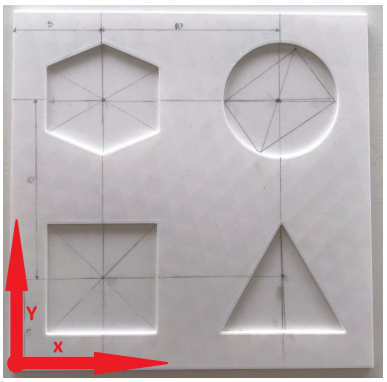
Figura 5.115: Definir WobjGrid en el FlexPendant.

Desde esta interfaz, seleccionar el método de los tres puntos para definir la base de coordenadas del objeto de trabajo escogido. Manualmente se tendrá que mover el robot para definir dos puntos en el eje X y uno en el eje Y. De esta manera, se definirá la base de coordenadas del objeto de trabajo en el entorno real.

Para este proyecto se han definido las bases de coordenadas de la cartulina y del tablero. La primera base se encuentra en el centro de la cartulina y hace referencia al objeto de trabajo "WobjGrid" [Figura 5.116a]. La segunda base se encuentra en la esquina inferior izquierda del tablero y hace referencia al objeto de trabajo "Tablero" [Figura 5.116b].



(a) Base de coordenadas del WobjGrid.



(b) Base de coordenadas del Tablero.

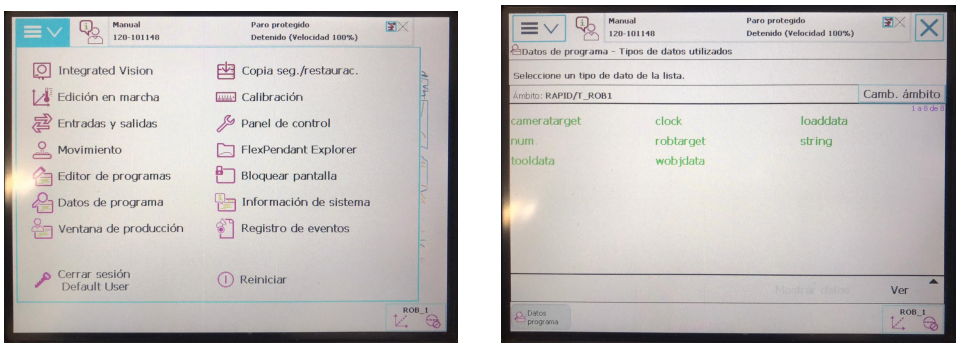
Figura 5.116: Objetos de trabajo definidos.

5.7.2.6. Definir los RobTargets

Los RobTargets son las posiciones definidas dentro del espacio de trabajo a las cuales el robot puede moverse.

- Para el programa en el entorno real se han definido cuatro RobTargets:
  1. RobTarget PASO: es un punto de paso entre la zona de detección de figuras y la zona de dejar las piezas.
  2. RobTarget HOME: es la posición inicial y final de la tarea que tiene que ejecutar el robot.
  3. RobTarget DejarPieza: es el punto que coincide con la base de coordenadas del objeto de trabajo "Tablero", desde este punto se define mediante la función "Offset" el punto de dejada de cada pieza en el tablero.
  4. RobTarget myrobtarget es el punto de cogida de pieza, en el que se define la altura de la pieza para que pueda ser manipulada en la zona de detección de figuras.

Para crear estas posiciones en el programa del entorno real, la única manera de hacerlo es desde el FlexPendant. Primeramente, se deberá acceder al "Menú" y seleccionar "Datos de programa" [Figura 5.117].



(a) Menú FlexPendant. **(b)** Datos de programa FlexPendant.

Figura 5.117: Interfaz del FlexPendant.

El siguiente paso es seleccionar ”robtarget”, desde esta ventana se podrán visualizar todas las posiciones definidas en el programa [Figura 5.118].

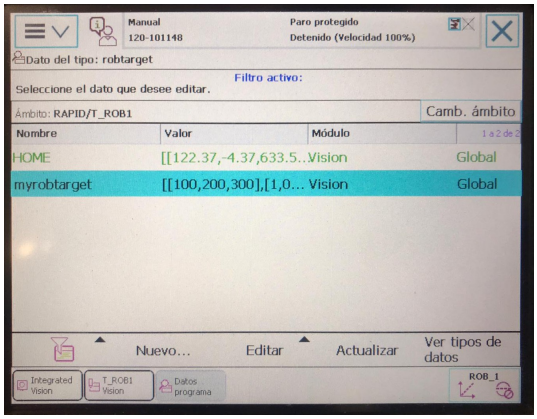


Figura 5.118: Ventana robtarget.

Para poder crear una nueva posición, seleccionar ”Nuevo”, aparecerá una ventana en la que se podrá especificar el nombre del robtarget, la base de coordenadas del objeto de trabajo al que está ligado y el módulo de programa en el que se va a inicializar [Figura 5.119].

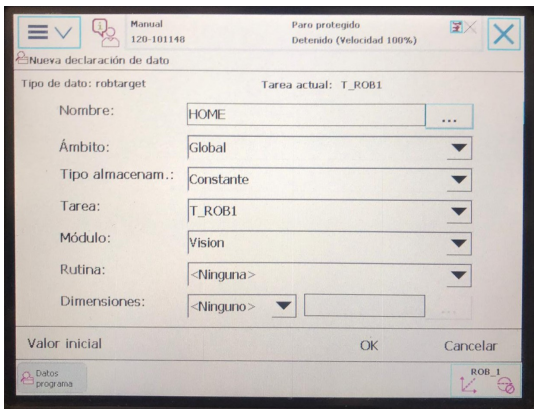


Figura 5.119: Crear un nuevo robtarget.

En cambio, si se desea modificar un robtarget creado previamente, solo se tiene que llevar el robot manualmente a la posición deseada especificando el objeto de trabajo por el cual se está moviendo y la herramienta de trabajo que está utilizando. Una vez el robot está en la posición deseada, hay que seleccionar el robtarget que requiere modificarse, seleccionar "Editar" y presionar la opción "Modificar posición" [Figura 5.120].

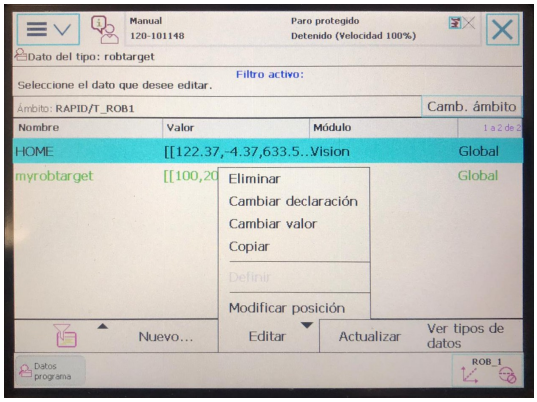


Figura 5.120: Modificar un robtarget.

5.8. Puesta en marcha

La manera en la que se ejecutan los programas tanto en el entorno virtual como en el entorno real son diferentes. Hay que tener en cuenta que la simulación no requiere de un controlador real, por lo que no es necesario utilizar los dispositivos físicos del laboratorio. En cambio, para poder ejecutar el programa que se ha realizado en el entorno real, es necesario poner en marcha todos los dispositivos asociados a la estación del laboratorio.

5.8.1. Puesta en marcha en simulación

Para ejecutar la simulación del programa realizado en el entorno virtual hay que dirigirse a la ventana "Simulación" y seleccionar la herramienta "Reproducir" [Figura 5.121].

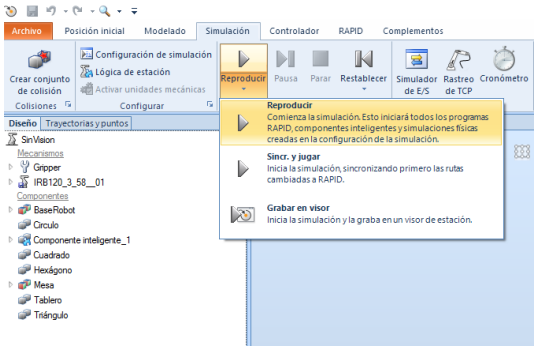


Figura 5.121: Ejecutar la simulación del programa en el entorno virtual.

5.8.2. Puesta en marcha en entorno real

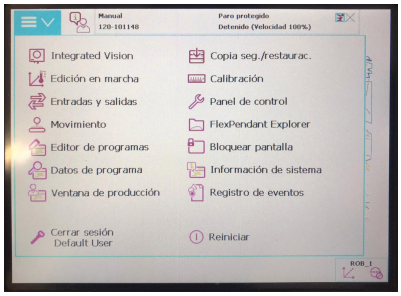
Para poner en marcha el robot en el entorno real y ejecutar el programa que se ha realizado primeramente se debe encender la alimentación del controlador [Figura 5.122a]. A continuación, colocar el robot en modo automático girando la llave en sentido antihorario y rearmar los motores presionando el botón blanco [Figura 5.122b].

(a) Encender la alimentación del controlador.

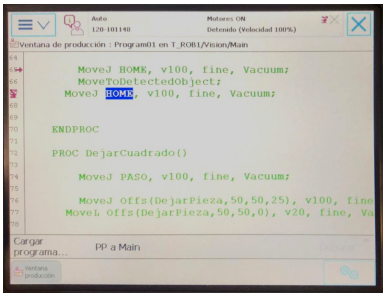
(b) Modo automático y rearme de motores.

Figura 5.122: Poner en marcha el sistema.

Posteriormente, hay que utilizar el FlexPendant para ejecutar el programa. Para ello, primero hay que dirigirse a la "Ventana de producción" y seleccionar el botón PP a main para que el programa se inicie desde la primera línea del código [Figura 5.123].



(a) Menú del FlexPendant.



(b) Ventana de producción.

Figura 5.123: Visualizar el programa cargado.

Finalmente, presionar el botón de Play del FlexPendant [Figura 5.124].

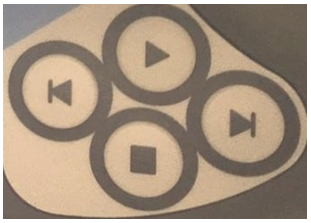


Figura 5.124: Botón Play.

6. Resultados

Para mostrar los resultados de los programas en el entorno de simulación y en el entorno real, se han grabado, editado y subido los comportamientos a Youtube. Para ambas partes los resultados han sido los esperados ya que han cumplido el objetivo de realizar la tarea de manipulación de piezas.

6.1. Entorno de simulación

Para visualizar el vídeo desde youtube seleccionar Video entorno simulación. También se puede acceder desde la URL: https://www.youtube.com/watch?v=S6nBwR6oJhw.

6.2. Entorno real

Para visualizar el vídeo desde youtube seleccionar Video entorno real. También se puede acceder desde la URL: https://www.youtube.com/watch?v=Ukpiy_UjOhA.

7. Conclusiones

Para concluir, se puede decir que se han cumplido los objetivos propuestos al principio del proyecto. Ya que se ha realizado una tarea de detección y manipulación de piezas sincronizando un sistema de visión junto con un sistema robótico en el entorno real. Además, se ha podido desarrollar una tarea de manipulación en el entorno de simulación, que ha llevado a una mejora de los conocimientos generales de la programación de robots industriales. De esta manera, se ha conseguido profundizar en el programa RobotStudio realizando ambos programas e integrando el sistema de visión. También se ha mejorado en el diseño de piezas 3D, creando tanto la herramienta de trabajo para el entorno de simulación como las figuras utilizadas en el entorno real. Y finalmente, se ha adquirido el conocimiento sobre el procedimiento a seguir para poner en marcha un sistema robótico real.

Bibliografía

ABB. (2004a). *Especificaciones del producto controller irc5 with flexpendant robotware 5.15.* Descargado de https://library.e.abb.com/public/2b5b950d68a0503cc1257c0c003cb703/3HAC041344-es.pdf

ABB. (2004b). *Especificaciones del producto controller software irc5.* Descargado de https://library.e.abb.com/public/de03bf63ea7379e3c1257c0c003b3a21/3HAC022349-es.pdf

ABB. (2008). *Manual del operador robotstudio.* Descargado de https://library.e.abb.com/public/6aeb483836740e11c1257b4b0052375b/3HAC032104-005_revE_es.pdf

ABB. (2009). *Manual del producto irc5 compact.* Descargado de https://abb.sluzba.cz/Pages/Public/IRC5RoboticsDocumentationRW6/Controllers/IRC5/IRC5%20Compact/es/3HAC047138-005.pdf

ABB. (2010). *Especificaciones del producto irb 120.* Descargado de https://search.abb.com/library/Download.aspx?DocumentID=3HAC035960-005&LanguageCode=es&DocumentPartId=&Action=Launch

ABB. (2018). *Manual de aplicaciones - visón integrada.* Descargado de https://www.edu.xunta.gal/centros/cafi/aulavirtual/pluginfile.php/44989/mod_folder/content/0/DocumentacionAlumno/Bibliografia/VisionIntegrada.pdf?forcedownload=1

Cognex. (s.f.). *Que es la herramienta patmax y por que funciona.* Descargado de https://www.cognex.com/es-es/blogs/machine-vision/what-is-the-patmax-tool-and-why-does-it-work

Cognex. (2008). *In-sight® serie 7000 sistema de visión manual de instalación.* Descargado de https://support.cognex.com/docs/is_590/ISE/ES/Manuals/is7000instES.pdf

Cortés, D. J. F. R. (2011). *Robótica - control de robots manipuladores.* Descargado de https://books.google.es/books?hl=es&lr=&id=cULVDQAAQBAJ&oi=fnd&pg=PT5&dq=historia+de+la+robotica+industrial&ots=LR-Equu1aW&sig=lUdg5SDD0bleaafjCqsG4xfGaV4#v=onepage&q&f=false

del Val Román, J. L. (2016). *Industria 4.0: la transformación digital de la industria.* Descargado de http://coddii.org/wp-content/uploads/2016/10/Informe-CODDII-Industria-4.0.pdf

Freecad getting started. (2007). Descargado de https://wiki.freecadweb.org/Getting_started

George devol. (1912-2011). Descargado de https://es.wikipedia.org/wiki/George_Devol

Joseph engelberger. (1925-1915). Descargado de https://es.wikipedia.org/wiki/Joseph_Engelberger

Raymond goertz. (1915-1970). Descargado de https://en.wikipedia.org/wiki/Raymond_Goertz

Victor scheinman. (1942-2016). Descargado de https://es.wikipedia.org/wiki/Victor_Scheinman

A. Código del módulo de programa de la simulación

Código A.1: Código del módulo Module1

```
MODULE Module1

   !Declarar los robot targets
   CONST robtarget Home:=[[424.975607337,0,559],
   [0.5,0,0.866025404,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

   CONST robtarget CogerCuadrado_10:=[[350,-200,2.5],
   [0,0,1,0],[-1,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

   CONST robtarget Paso:=[[414.533061312,-8.368695842,361.501766221],
   [0.309020979,-0.219848367,0.921102615,-0.087992629],
   [-1,-1,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

   CONST robtarget CogerTriangulo_10:=[[225,200,2.5],
   [0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

   CONST robtarget CogerHexagono_10:=[[350,200,2.5],
   [0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

   CONST robtarget CogerCirculo_10:=[[225,-200,2.5],
   [0,0,1,0],[-1,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

   CONST robtarget DejarFiguras_10:=[[300,-100,5],
   [0,0,1,0],[-1,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

   PROC main()

      Reset Vacio;
      CogerCuadrado;
      DejarCuadrado;
      CogerTriangulo;
      DejarTriangulo;
      CogerCirculo;
      DejarCirculo;
      CogerHexagono;
      DejarHexagono;

   ENDPROC

```

```
40  PROC CogerCuadrado()
41
42      !Mover el robot a la posición inicial
43      MoveJ Home,v100,z0,Vacuum\WObj:=wobj0;
44
45      !Mover el robot a la posición de coger pieza
46      MoveJ Offs(CogerCuadrado_10,0,0,+50),v100,z0,Vacuum\WObj:=WO_Cuadrado;
47      MoveL CogerCuadrado_10,v20,z0,Vacuum\WObj:=WO_Cuadrado;
48
49      !Activar la ventosa
50      WaitTime 0.5;
51      SetDO Vacio,1;
52      WaitTime 0.5;
53
54      !Mover el robot a la posición de aproximación
55      !y después al punto de paso
56      MoveL Offs(CogerCuadrado_10,0,0,+50),v20,z0,Vacuum\WObj:=WO_Cuadrado;
57      MoveJ Paso,v1000,z100,Vacuum\WObj:=wobj0;
58
59  ENDPROC
60
61  PROC CogerTriangulo()
62      MoveJ Home,v100,z0,Vacuum\WObj:=wobj0;
63      MoveJ Offs(CogerTriangulo_10,0,0,+50),v100,z0,Vacuum\WObj:=WO_Triangulo;
64      MoveL CogerTriangulo_10,v20,z0,Vacuum\WObj:=WO_Triangulo;
65      WaitTime 0.5;
66      SetDO Vacio,1;
67      WaitTime 0.5;
68      MoveL Offs(CogerTriangulo_10,0,0,+50),v20,z0,Vacuum\WObj:=WO_Triangulo;
69      MoveJ Paso,v1000,z100,Vacuum\WObj:=wobj0;
70  ENDPROC
71
72  PROC CogerHexagono()
73      MoveJ Home,v100,z0,Vacuum\WObj:=wobj0;
74      MoveJ Offs(CogerHexagono_10,0,0,50),v100,z0,Vacuum\WObj:=WO_Hexagono;
75      MoveL CogerHexagono_10,v20,z0,Vacuum\WObj:=WO_Hexagono;
76      WaitTime 0.5;
77      SetDO Vacio,1;
78      WaitTime 0.5;
79      MoveL Offs(CogerHexagono_10,0,0,50),v20,z0,Vacuum\WObj:=WO_Hexagono;
80      MoveJ Paso,v1000,z100,Vacuum\WObj:=wobj0;
81  ENDPROC
82
83  PROC CogerCirculo()
84      MoveJ Home,v100,z0,Vacuum\WObj:=wobj0;
85      MoveJ Offs(CogerCirculo_10,0,0,50),v100,z0,Vacuum\WObj:=WO_Circulo;
86      MoveL CogerCirculo_10,v20,z0,Vacuum\WObj:=WO_Circulo;
87      WaitTime 0.5;
88      SetDO Vacio,1;
89      WaitTime 0.5;
90      MoveL Offs(CogerCirculo_10,0,0,50),v20,z0,Vacuum\WObj:=WO_Circulo;
```

```
91          MoveJ Paso,v1000,z100,Vacuum\WObj:=wobj0;
92      ENDPROC
93
94      PROC DejarTriangulo()
95          !Mover el robot a la zona de dejar figura
96          MoveJ Offs(DejarFiguras_10,45,50,50),v100,z0,Vacuum\WObj:=WO_Tablero;
97          MoveL Offs(DejarFiguras_10,45,50,0),v20,z0,Vacuum\WObj:=WO_Tablero;
98
99          !Desactivar la ventosa
100         WaitTime 0.5;
101         SetDO Vacio,0;
102         WaitTime 0.5;
103
104         !Mover el robot a la posición de aproximación de dejada de figura
105         !y después a la posición de paso
106         MoveL Offs(DejarFiguras_10,45,50,50),v20,z0,Vacuum\WObj:=WO_Tablero;
107         MoveJ Paso,v1000,z100,Vacuum\WObj:=wobj0;
108     ENDPROC
109
110     PROC DejarCuadrado()
111         MoveJ Offs(DejarFiguras_10,50,150,50),v100,z0,Vacuum\WObj:=WO_Tablero;
112         MoveL Offs(DejarFiguras_10,50,150,0),v20,z0,Vacuum\WObj:=WO_Tablero;
113         WaitTime 0.5;
114         SetDO Vacio,0;
115         WaitTime 0.5;
116         MoveL Offs(DejarFiguras_10,50,150,50),v20,z0,Vacuum\WObj:=WO_Tablero;
117         MoveJ Paso,v1000,z100,Vacuum\WObj:=wobj0;
118     ENDPROC
119
120     PROC DejarHexagono()
121         MoveJ Offs(DejarFiguras_10,150,150,50),v100,z0,Vacuum\WObj:=WO_Tablero;
122         MoveL Offs(DejarFiguras_10,150,150,0),v20,z0,Vacuum\WObj:=WO_Tablero;
123         WaitTime 0.5;
124         SetDO Vacio,0;
125         WaitTime 0.5;
126         MoveL Offs(DejarFiguras_10,150,150,50),v20,z0,Vacuum\WObj:=WO_Tablero;
127         MoveJ Paso,v1000,z100,Vacuum\WObj:=wobj0;
128     ENDPROC
129
130     PROC DejarCirculo()
131         MoveJ Offs(DejarFiguras_10,150,50,50),v100,z0,Vacuum\WObj:=WO_Tablero;
132         MoveL Offs(DejarFiguras_10,150,50,0),v20,z0,Vacuum\WObj:=WO_Tablero;
133         WaitTime 0.5;
134         SetDO Vacio,0;
135         WaitTime 0.5;
136         MoveL Offs(DejarFiguras_10,150,50,50),v20,z0,Vacuum\WObj:=WO_Tablero;
137         MoveJ Paso,v1000,z100,Vacuum\WObj:=wobj0;
138     ENDPROC
139
140 ENDMODULE
```

B. Código del módulo de programa del entorno real

Código B.1: Código del módulo Visión

```
MODULE Vision
   !Declarar el trabajo de visión que se va a utilizar en el programa
   CONST string myjob := "Figuras3.job";
   VAR cameratarget mycameratarget;

   !Declarar herramienta de trabajo
   PERS tooldata Vacuum:=[TRUE,[[0,0,60],
   [1,0,0,0]],[0.3,[0,0,1],[1,0,0,0],0,0,0]];

   !Declarar los objetos de trabajo
   TASK PERS wobjdata WobjGrid2:=[FALSE,TRUE,"",[[426.165,163.188,22.9781],
   [0.708151,-0.00707719,-0.00616661,-0.705998]],[[5.91058,9.13198,0],
   [0.999839,0,0,0.0179505]]];

   TASK PERS wobjdataTablero:=[FALSE,TRUE,"",[[326.303,-22.277,32.1978],
   [0.708382,-1.18486E-05,2.39246E-05,-0.705829]],[[0,0,0],[1,0,0,0]]];

   !Declarar los robot target
   CONST robtarget HOME:=[[136.59,-0.20,498.47],
   [0.00818261,0.904813,-0.425624,0.00955078],[0,-1,2,0],
   [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

   CONST robtarget myrobtarget:=[[-0.05,2.65,2.83],
   [0.0048476,-0.948875,-0.31559,-0.00407687],[0,-1,2,0],
   [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

   CONST robtarget DejarPieza:=[[-0.27,-0.56,-0.71],
   [0.00115169,-0.94137,-0.337146,-0.0123687],[-1,-1,2,0],
   [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

   CONST robtarget PASO:=[[405.44,17.41,114.63],
   [0.00812611,0.904814,-0.425621,0.00966605],[0,-1,2,0],
   [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

   PROC MoveToDetectedObject()
      !Seleccionar y activar la cámara que tenemos
      !conectada con el trabajo de visión
      CamSetProgramMode CamaraRobot2;
      CamLoadJob CamaraRobot2, myjob;
```

```
40      CamSetRunMode CamaraRobot2;
41
42      !Adquirir una imagen de la zona de detección de figuras y guardarla
43      CamReqImage CamaraRobot2;
44      CamGetResult CamaraRobot2, mycameratarget;
45      WobjGrid2.oframe := mycameratarget.cframe;
46
47      !Mover el robot hacia la zona de aproximación de
48      !manipulación de la figura
49      MoveL Offs(myrobtarget,0,0,50), v100, fine, Vacuum\WObj:=WobjGrid2;
50      !Mover el robot hacia la zona de manipulación de la figura
51      MoveL myrobtarget, v100, fine, Vacuum\WObj:=WobjGrid2;
52
53      !Activar la aspiración de la ventosa
54      WaitTime 0.5;
55      SetDo Soplado,1;
56      WaitTime 0.5;
57
58      !Dependiendo de la figura detectada llamar a la función
59      !que las deja en su posición correspondiente
60      IF mycameratarget.name = "Circulo" THEN
61          DejarCirculo;
62      ELSEIF mycameratarget.name = "Cuadrado" THEN
63          DejarCuadrado;
64      ELSEIF mycameratarget.name = "Poligono" THEN
65          DejarHexagono;
66      ELSEIF mycameratarget.name = "Triangulo" THEN
67          DejarTriangulo;
68      ENDIF
69  ENDPROC
70
71  PROC Main()
72      !Mover el robot a la posición inicial HOME
73      MoveJ HOME, v100, fine, Vacuum;
74
75      !Llamar a la función que adquiere una imagen de la zona de
76      !detección y manipula las piezas
77      MoveToDetectedObject;
78
79      !Mover el robot a la posición final HOME
80      MoveJ HOME, v100, fine, Vacuum;
81  ENDPROC
82
83  PROC DejarCuadrado()
84      !Mover el robot a la posición de paso entre
85      !la zona de detección y el tablero
86      MoveJ PASO, v100, fine, Vacuum;
87
88      !Mover el robot hacia la zona de aproximación de dejar la figura
89      MoveJ Offs(DejarPieza,50,50,25), v100, fine, Vacuum\WObj:=Tablero;
90
```

```
91        !Mover el robot hacia la zona de dejar la figura
92        MoveL Offs(DejarPieza,50,50,0), v20, fine, Vacuum\WObj:=Tablero;
93
94        !Desactivar la aspiración de la ventosa
95        WaitTime 0.5;
96        SetDo Soplado,0;
97        WaitTime 0.5;
98
99        !Mover el robot hacia la zona de aproximación de dejar la figura
100       MoveL Offs(DejarPieza,50,50,25), v100, fine, Vacuum\WObj:=Tablero;
101   ENDPROC
102
103   PROC DejarCirculo()
104       MoveJ PASO, v100, fine, Vacuum;
105       MoveJ Offs(DejarPieza,150,150,25), v100, fine, Vacuum\WObj:=Tablero;
106       MoveL Offs(DejarPieza,150,150,0), v20, fine,Vacuum\WObj:=Tablero;
107
108       WaitTime 0.5;
109       SetDo Soplado,0;
110       WaitTime 0.5;
111
112       MoveL Offs(DejarPieza,150,150,25), v100, fine, Vacuum\WObj:=Tablero;
113   ENDPROC
114
115   PROC DejarTriangulo()
116       MoveJ PASO, v100, fine, Vacuum;
117
118       MoveJ Offs(DejarPieza,150,50,25), v100, fine, Vacuum\WObj:=Tablero;
119       MoveL Offs(DejarPieza,150,50,0), v20, fine, Vacuum\WObj:=Tablero;
120
121       WaitTime 0.5;
122       SetDo Soplado,0;
123       WaitTime 0.5;
124
125       MoveL Offs(DejarPieza,150,50,25), v100, fine, Vacuum\WObj:=Tablero;
126   ENDPROC
127
128   PROC DejarHexagono()
129       MoveJ PASO, v100, fine, Vacuum;
130
131       MoveJ Offs(DejarPieza,50,150,25), v100, fine, Vacuum\WObj:=Tablero;
132       MoveL Offs(DejarPieza,50,150,0), v20, fine, Vacuum\WObj:=Tablero;
133
134       WaitTime 0.5;
135       SetDo Soplado,0;
136       WaitTime 0.5;
137
138       MoveL Offs(DejarPieza,50,150,25), v100, fine, Vacuum\WObj:=Tablero;
139   ENDPROC
140ENDMODULE
```